2006-03-01

Using Case-based Reasoning for Spam Filtering

Sarah Jane Delany
Technological University Dublin, sarahjane.delany@tudublin.ie

Using Case-Based Reasoning for Spam Filtering

Sarah Jane Delany

A thesis submitted to the Dublin Institute of Technology

in fulfillmeut of the requirements for the degree of

Doctor of Philosophy

School of Computing

March 2006

Declaration

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The work reported on in this thesis conforms to the principles and requirements of the Institute's guidelines for ethics in research.

The Institute has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis by duly acknowledged.

Signature Sarah Jane Delany Date 22/6/06

Acknowledgements

I would like to express my gratitude to my supervisors, Brendan O'Shea, the Head of the School of Computing for his support and encouragement and Barry Smyth, my co-supervisor for his invaluable advise and help.

I would also like to thank my colleagues who work with me in the School of Computing in Kevin Street. Their friendship and support was great. Many thanks also go to all the members of the Machine Learning Group in Trinity College Dublin with whom I spent almost two years, in particular, Lorcan Coyle, Dónal Doyle, Alexey Tsymbal and Anton Zamolotskikh with whom I worked closely. The group was a stimulating and enjoyable place to work and I learned a lot from my time there. Thanks especially goes to Mike Carney, Pádraig Cunningham, Kuda Dube, Al Gray, Marco Grimaldi, Lucy Hederman, Conor Nugent, Oliver O'Connor and Ciarán O'Leary all of whom installed and ran the spam filter for me.

Finally, I owe a debt of gratitude to my family who have supported me throughout the process. Thanks to Pat, Peter and Joanne for stepping into the breach on many occasions, to Ailbhe, Eoghan and Aoife for their patience while their mother worked too many hours and most especially to Pádraig for his support, advise and love without which this would not have been possible.

Sarah Jane Delany

Dublin Institute of Technology

March 2006

Abstract

Spam is a universal problem with which everyone is familiar. Figures published in 2005 state that about 75% of all email sent today is spam. In spite of significant new legal and technical approaches to combat it, spam remains a big problem that is costing companies meaningfnl amounts of money in lost productivity, clogged email systems, bandwidth and technical support.

A number of approaches are used to combat spam including legislative measures, authentication approaches and email filtering. The most common filtering technique is content-based filtering which uses the actual text of the message to determine whether it is spam or not.

One of the main challenges of content based spam filtering is concept drift; the concept or the characteristics used by the filter to identify spam email are constantly changing over time. Concept drift is very evident in email and spam, in part due to the arms race that exists between the spammers and the filter producers. The spammers continually change the content and structure of the spam emails as the filters are modified to catch them.

In this thesis we present *Email Classification Using Examples* (ECUE) a content based approach to spam filtering that can handle the concept drift inherent in spam email. We apply the machine learning technique of case-based reasoning which models the emails as cases in a knowledge-base or case-base. The approach used in ECUE involves two components; a case-base editing stage and a case-base update policy. We present a new technique for case-base editing called *Competence-Based Editing* which uses the competence properties of the cases in the case-base to determine which cases are harmful to the predictive power of the case-base and should be removed. The update policy allows new examples of spam and legitimate emails to be added to the case-base as they are encountered allowing ECUE to track the concept drift.

We compare the case-based approach to an ensemble approach which is a more standard technique for handling concept drift and present a prototype email filtering applica-

tion that demonstrates how the ECUE approach to spam filtering can handle the concept drift.

Contents

List of Figures

List of Tables

Associated Publications

The publications that are related to this thesis are listed below:

1. Pádraig Cunningham, Niamh Nowlan, Sarah Jane Delany and Mads Haahr (2003) A Case-based Approach to Spam Filtering that can track Concept Drift. In: ICCBR 2003 Workshop on Long-Lived CBR Systems,

2. Sarah Jane Delany and Pádraig Cunningham (2004) An Analysis of Case-base Editing in a Spam Filtering System. In: P. Funk & P.A. Gonzales Calero (eds.), Advances in Case-Based Reasoning, Proceedings of Seventh European Conference on Case-Based Reasoning), LNAI 3155 pp.128–141 Springer Verlag,

3. Sarah Jane Delany, Pádraig Cunningham and Lorcan Coyle (2004) An Assessment of Case-Based Reasoning for Spam Filtering. In L. McGinty & B. Crean (eds.) Proceedings of the Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2004), pp 9–18,

4. Sarah Jane Delany, Pádraig Cunningham, Alexey Tsymbal and Lorcan Coyle (2004) A Case-Based Technique for Tracking Concept Drift in Spam Filtering, In: A. Macintosh, R. Ellis & T. Allen (eds.) Applications and Innovations in Intelligent Systems XII, Procs. of AI 2004, p3–16, Springer,

5. Sarah Jane Delany, Pádraig Cunningham and Lorcan Coyle (2005) An Assessment of Case-Based Reasoning for Spam Filtering, Artificial Intelligence Review 24(3–4) p359–378, Springer,

6. Sarah Jane Delany, Pádraig Cunningham, Alexey Tsymbal and Lorcan Coyle (2005) A Case-Based Technique for Tracking Concept Drift in Spam Filtering, Knowledge Based Systems, 18 (4–5) p187–195, Elsevier,

7. Sarah Jane Delany, Pádraig Cunningham, Dónal Doyle, Anton Zamolotskikh (2005) Generating Estimates of Classification Confidence for a Case-Based Spam Filter In:

Proceedings of Sixth International Conference on Case-Based Reasoning, p170–190, Springer,

8. Matt Healy, Sarah Jane Delany and Anton Zamolotskikh (2005) An Assessment of CBR for Short Text Message Classification In: N. Creaney (ed.) Proceedings of the Sixteenth Irish Conference on Articial Intelligence and Cognitive Science (AICS 2005), p257–266,

9. Anton Zamolotskikh, Sarah Jane Delany and Pádraig Cunningham (2006) A methodology for comparing classifiers that allow the control of bias, In: Proceedings of 21^{st} ACM Symposium on Applied Computing, p582–587, ACM, New York.

Paper 5 is an extended verion of paper 3 which was accepted for publication in a Special Issue of the Artifical Intelligence Review journal. Also, paper 4 won the Best Application Paper award at the AI-2004 conference and as a result was published as paper 6 in a Special Issue of the Knowledge Based Systems journal.

Chapter 1

INTRODUCTION

One of the most significant impacts of the Internet has been the provision of email, a simple and effective means of communication to anyone and everyone world wide. However the simplicity and ease-of-use of email is being threatened by the sheer volume of unsolicited emails known as spam. Spam email is a problem that almost everyone is familiar with. Most people can identify with the person whose daily emails suggest that they are ill or impotent, in need of cheap prescription drugs, that they have lots of spare cash to spend on endless investment opportunities or that they have no friends and are constantly looking to meet people online.

There are a variety of ways to try to stop or reduce the amount of spam an individual receives. These include legislative measures with significant anti-spam laws being introduced world-wide over the last couple of years. Other techniques known as authentication techniques place the emphasis on 'you proving who you are' before email will be accepted from you. But the most common techniques are filtering techniques, attempting to identify from the content or other characteristics of the message that it is spam. In spite of the number of spam combatting techniques available, the volumes of spam on the Internet still seem to be rising.

A key challenge facing any computer system designed to cope with spam is the fact that spam is constantly changing. There are a number of factors that contribute to this, for example new opportunities are constantly being exploited by spammers and seasonal effects such as the advertising of weight loss products after Christmas, have an impact. A key factor, though, is that a virtual arms race exists between the spammers and the filter producers. The spammers continually change the content and structure of the spam emails as the filters are modified to catch them. This is an example of 'concept drift' as the spam 'concept' changes over time and the characteristics used by the system to identify

spam are therefore constantly changing.

There have been a variety of machine learning techniques applied to the problem of spam filtering. The main technique used is Naïve Bayes, a probabilistic classifier that has been shown to be good at text classification (Lewis and Ringuette 1994, Lewis 1998). Support Vector Machines (Joachims 1998, Dumais et al. 1998), also good at text classification, have also been used. Both these techniques are categorised as *eager* learners, as the system is trained in advance of receiving any requests to process. Eager learners construct a *model* from the training examples and then use the model itself to process the requests. *Lazy* learners (Aha 1997), on the other hand, do not build a model. They select an appropriate subset of training data when an explicit request is received and use this to process the request.

Lazy learners offer some advantages over eager learners especially in changing environments such as spam filtering. They facilitate updates to the training data allowing the introduction of new examples to the learning process to help to track the concept drift. As they are also local learners (Bottou and Vapnik 1992), using a appropriate subset of the training data to process each request, they are suited to non homogeneous domains such as spam. In this thesis our hypothesis is that a lazy learner such as Case-Based Reasoning can handle the concept drift in spam email.

Case-Based Reasoning (CBR) is a machine learning technique that uses or adapts the solutions from previously solved problems to solve new problems (Riesbeck and Shank 1989). Previous experiences are stored as cases in a knowledge or case-base. In a text classification problem such as spam filtering the cases are represented as n-dimensional vectors of terms, each term identified by the parsing or tokenising of the text in the document. Each vector or case also includes the classification of the document, i.e. spam or nonspam.

Two main challenges face the application of CBR to spam filtering. Firstly how to manage the training data, the vast collection of emails that exist for most users of email and secondly, how to handle concept drift. The issue of managing the volume of training data requires a case-base editing policy that selects those training examples that are better at prediction than others. Handling concept drift requires a new case selection policy to allow the system to learn the changes in the underlying phenomenon from new examples of spam and legitimate email.

Our approach, Email Classification Using Examples (ECUE), involves a two-step case-base management policy including:

(i) a case-base editing technique called Competence Based Editing (CBE) and

(ii) a case-base update policy.

CBE is a novel competence-based procedure that includes two stages, a noise reduction phase called Blame Based Noise Reduction (BBNR) and a redundancy elimination phase called Conservative Redundancy Reduction (CRR). BBNR focuses on the damage that certain cases are causing in classifications and is effective in spam filtering as it identifies and removes the cases in the case-base that cause other cases to be misclassified rather than the more common technique of removing cases that are actually misclassified. CRR was motivated by the observation that state-of-the-art techniques were inclined to be too aggressive in removing cases and tended to result in some loss of generalisation accuracy, at least in the domain of spam filtering. We will show in this thesis how our technique CBE can assist in managing the training data and also produces reduced case-bases that have the best generalisation accuracy in this domain, as compared to other popular case-base editing techniques.

Our case-base update policy centers on two hierarchical levels of learning. The first and simplest level is a continuous case-base update with training examples that have been misclassified by our filter. The second level is a periodic retraining of the classifier to reselect features that may be more predictive of spam and legitimate email. We will show how this update policy combined with the initial CBE case-editing procedure can effectively handle the concept drift that is so evident in the spam filtering domain.

An analysis of the machine learning literature on concept drift suggests that there are three general approaches to tracking concept drift; instance selection, instance weighting and ensemble learning (Tsymbal 2004). Research shows that ensemble approaches are among the most effective techniques (Kolter and Maloof 2003, Kuncheva 2004). Our approach, ECUE, falls into the category of instance selection, the goal of which is to select instances (training examples) that are representative of the current concept. In this thesis we compare our instance selection approach with the more common ensemble approach and show that our instance selection approach is more straightforward and as effective as the ensemble approaches that we evaluate.

A learning system such as ECUE (which is capable of handling concept drift) will need user intervention to indicate when mistakes have been made to allow the update policy to be triggered. This places quite an onus on the user of the system to check the predictions made by the system for mistakes. Identifying missed spam is relatively straighforward

as they will show up in the user's Inbox, as ECUE assumes that they are legitimate emails. Identifying legitimate email incorrectly classified as spam is more tedious. This usually involves periodically checking through a folder containing all email identified as spam by the filter. If an estimate of the prediction confidence can be produced for every spam prediciton it may remove some of the onus on the user of checking all the system's predictions; the user being able to ignore those predictions that are made with confidence. In this thesis we also discuss our approach to generating predictions of confidence.

1.1 Contributions of this Thesis

This thesis is concerned with the application of instance-based learning to the problem of spam filtering and most specifically to the problem of tracking concept drift in spam and legitimate emails. The main contributions of this thesis are the following:

- The implementation of a spam filter application that learns from new examples of spam and legitimate email,
- The development of a case-base maintenance policy including case-base editing and case-base update that handles concept drift iu spam,
- The development of a noise reduction algorithm called Blame Based Noise Reduction (BBNR) which effectively removes noisy and exceptional cases from a case-base of spam and legitimate emails,
- The application of BBNR in a new case-base editing technique called Competence Based Editing (CBE),
- An evaluation that shows that the case-base maintenance policy is better at handling concept drift in spam and legitimate emails than an ensemble approach,
- The development of a confidence measure that reduces the effort involved iu checking for false positives (i.e. legitimate emails that have been incorrectly classified as spam by a spam filter).

1.2 Summary and Structure of this Thesis

The next chapter, Chapter 2, discusses the problem of spam and the variety of ways of combatting it with particular emphasis on filtering techniques. This chapter examines

existing research which has applied machine learning techniques to spam filtering and highlights the main challenges of applying such techniques to the problem of spam filtering.

Chapter 3 describes the process of Case-Based Reasoning with particular emphasis on the challenges facing the application of CBR to spam filtering. It discusses associated research areas including case-base editing techniques which can assist in managing the training data and approaches for handling concept drift, a problem any learning system for spam filtering will face.

Chapter 4 describes ECUE, our system for spam filtering that can handle concept drift discussing the design decisions made in relation to feature extraction, feature selection and case representation and how case retrieval and case-base editing are performed. This chapter also explains the design of the prototype system, which integrates with email client software, that was implemented to allow an evalution of ECUE in a real-time online setting.

Chapter 5 discusses the evaluation of ECUE. It describes the two types of evaluations performed on the system. It discusses the offline evaluations on a number of email datasets that support a number of conclusions including, among others, ECUE's ability to handle concept drift. This chapter also discusses the online evaluation of the system in a live spam filtering situation.

Chapter 6 compares ECUE's case-based approach to handling concept drift to the more common approach of using ensembles of classifiers while Chapter 7 discusses our approach to generating estimates of classification confidence when predicting spam. Finally Chapter 8 concludes the thesis and describes some further work that could be investigated.

Chapter 2

SPAM

The first spam email, see Figure 2.1, is believed to have been sent in May 1978 by a DEC marketing representative named Gary Thuerk to a large number of Arpanet[1] users along the west coast of the U.S. inviting them to a product presentation[2]. Arpanet user addresses were published at that time in a printed directory and Thuerk used this to get the list of recepients. In those days the Arpanet had an official use policy which restricted its use to the support of education and research and this was a clear violation of this policy. Figure 2.2 shows the reaction from the Defense Communications Agency (DCA) part of the U.S. Department of Defence who ran Arpanet. Thuerk's boss also received a call from the DCA complaining about Thuerk's message.

The term *spam* was originally associated with bulk, unwanted messages posted to Usenet newsgroups[3]. The term is believed to originate from the MUD (multi-user-dungeon) community, a real-time multi-person shared environment/game. Within the MUD community the term *spam* was applied to either flooding a computer with too much data causing it to crash or flooding a chat session with automated text/files (rather than typing yonr own input). There is also a body of opinion that believes the term *spam* came from a Monty Python sketch[4] of a couple trying to order a meal with nothing but spam, spam and more spam on the menu.

One of the first mass mailings on Usenet was from Clarence L. Thomas IV in January 1994, who multi-posted a long religious message entitled *Global Alert for All: Jesus is Coming Soon* about the end of the world to all newsgroups[5]. However, the advent of

[1] Advanced Research Projects Agency Network (Arpanet) was the world's first packet-switching network, in effect what the Internet was originally called.

[2] www.mailmsg.com/SPAM_history_003.htm

[3] www.templetons.com/brad/spamterm.html

[4] www.detritus.org/spam/skit.html

[5] groups.google.com/group/sci.stat.edu/msg/8cb0e6b6941bac09

DIGITAL WILL BE GIVING A PRODUCT PRESENTATION OF THE NEWEST MEMBERS OF THE
DECSYSTEM-20 FAMILY; THE DECSYSTEM-2020, 2020T, 2060, AND 2060T. THE
DECSYSTEM-20 FAMILY OF COMPUTERS HAS EVOLVED FROM THE TENEX OPERATING SYSTEM
AND THE DECSYSTEM-10 <PDP-10> COMPUTER ARCHITECTURE. BOTH THE DECSYSTEM-2060T
AND 2020T OFFER FULL ARPANET SUPPORT UNDER THE TOPS-20 OPERATING SYSTEM.
THE DECSYSTEM-2060 IS AN UPWARD EXTENSION OF THE CURRENT DECSYSTEM 2040
AND 2050 FAMILY. THE DECSYSTEM-2020 IS A NEW LOW END MEMBER OF THE
DECSYSTEM-20 FAMILY AND FULLY SOFTWARE COMPATIBLE WITH ALL OF THE OTHER
DECSYSTEM-20 MODELS.

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY
AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS
MONTH. THE LOCATIONS WILL BE:

TUESDAY, MAY 9, 1978 - 2 PM
HYATT HOUSE (NEAR THE L.A. AIRPORT)
LOS ANGELES, CA

THURSDAY, MAY 11, 1978 - 2 PM
DUNFEY'S ROYAL COACH
SAN MATEO, CA
(4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER
DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND,
PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE
FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY

Figure 2.1: The First Spam Message sent by a Gary Thuerk, a DEC marketing representative, to a huge number of Arpanet addresses on the west coast of the U.S.

commercial spam as we now know it started later in 1994 when two lawyers, Cantel and Siegel, posted a message to 6000 newsgroups advertising a Green Card Lottery[6], see Figure 2.3. They continued posting for some time and reputedly made some money from their efforts.

Spam has been variously defiued as the use of any electronic connnunications medium to send unsolicited messages to someone in bulk[7], or 'unsolicted bulk email' (UBE)[8] or bulk email from a stranger[9]. The main common theme appears to be that unsolicited, unwanted email from someone you don't know is spam, whether it is sent in bulk or not. However, there does need to be nothing specific about you as the recipient of the message for it to be spam, for instance, a message from a student that you don't know looking for a postgraduate position, is not necessarily spam.

Spam has become a universal problem; everyone is familiar with it. Figures published in 2005 by online magaziue TechWorld[10], based on an average of the published figures from a number of the larger anti-spam software vendors, state that about 75% of all email sent today is spam[11]. In fact, oue email filtering company, Email Systems, have found

[6] www.mailmsg.com/SPAM_history_001.htm
[7] www.wikipedia.org/
[8] www.spamhaus.org/
[9] www.templetons.com/brad/spam/define.html
[10] www.techworld.com/
[11] www.techworld.com/security/features/index.cfm?featureid=1372

ON 2 MAY 78 DIGITAL EQUIPMENT CORPORATION (DEC) SENT OUT AN ARPANET MESSAGE ADVERTISING THEIR NEW COMPUTER SYSTEMS. THIS WAS A FLAGRANT VIOLATION OF THE USE OF ARPANET AS THE NETWORK IS TO BE USED FOR OFFICIAL U.S. GOVERNMENT BUSINESS ONLY. APPROPRIATE ACTION IS BEING TAKEN TO PRECLUDE ITS OCCURRENCE AGAIN.

IN ENFORCEMENT OF THIS POLICY DCA IS DEPENDENT ON THE ARPANET SPONSORS, AND HOST AND TIP LIAISONS. IT IS IMPERATIVE YOU INFORM YOUR USERS AND CONTRACTORS WHO ARE PROVIDED ARPANET ACCESS THE MEANING OF THIS POLICY.

THANK YOU FOR YOUR COOPERATION.

MAJOR RAYMOND CZAHOR

CHIEF, ARPANET MANAGEMENT BRANCH, DCA

Figure 2.2: Reaction from the Defense Communications Agency (DCA) part of the U.S. Department of Defence who ran Arpanet, to the first spam message

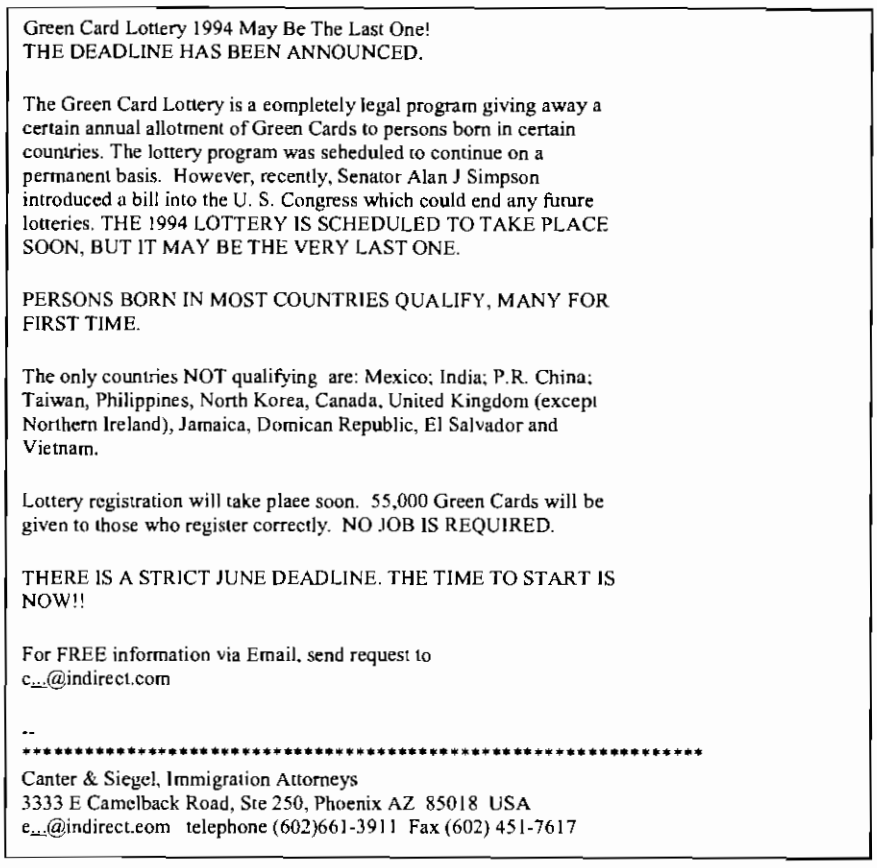
that on average spam email accounts for over 90% of the email that it filters daily[12].

Sending spam is cheap, with a simple dial-np connection and a PC a spammer can send hundreds of emails an honr. The main cost of spam is borne by the receiver, usually the corporation or ISP that is managing the mailboxes to which the spam is sent, however there is a significant implicit cost to the email user who wastes time reading and deleting the email from their mailboxes. Spammers receive commission on each referral that is received in response to their spam message. In a direct marketing campaign a return of 2% is usually reqnired to break even (Ferreri 1999), however spammers can break even with a response rate as low as 0.001% (Judge et al. 2005). Spammers collect email addresses using a variety of methods including harvesting software which searches the Internet for email addresses, dictionary attacks where email addresses are guessed using common words from a dictionary, or by buying lists from list brokers. A list of 100 million email address can be purchased for as little as $29.95.[13].

Technology is on the side of the spammer as it is getting increasingly cheaper to send email and increasingly more difficult to ensure your computer is one hundred percent spam-safe. Spammers also use people's ignorance of the necessary anti-virus measures to comandeer computer systems and use them to transmit spam without the owner of the system realising. Such compromised systems are known as zombies.

In spite of significant new legislative and technical measures to combat it, spam remains a big problem that is costing companies significant amounts of money in lost productivity, clogged email systems, bandwidth and technical support. Ferris Research Inc[14], a market

[12] www.emailsystems.com/news.php?itemid=219
[13] www.mailmillions.com
[14] www.ferris.com/



Green Card Lottery 1994 May Be The Last One!
THE DEADLINE HAS BEEN ANNOUNCED.

The Green Card Lottery is a eompletely legal program giving away a certain annual allotment of Green Cards to persons born in certain countries. The lottery program was seheduled to continue on a permanent basis. However, recently, Senator Alan J Simpson introduced a bill into the U. S. Congress which could end any future lotteries. THE 1994 LOTTERY IS SCHEDULED TO TAKE PLACE SOON, BUT IT MAY BE THE VERY LAST ONE.

PERSONS BORN IN MOST COUNTRIES QUALIFY, MANY FOR FIRST TIME.

The only countries NOT qualifying are: Mexico; India; P.R. China; Taiwan, Philippines, North Korea, Canada, United Kingdom (excepl Northern Ireland), Jamaica, Domican Republic, El Salvador and Vietnam.

Lottery registration will take plaee soon. 55,000 Green Cards will be given to those who register correctly. NO JOB IS REQUIRED.

THERE IS A STRICT JUNE DEADLINE. THE TIME TO START IS NOW!!

For FREE information via Email, send request to
c...@indirect.com

--
****
Canter & Siegel, Immigration Attorneys
3333 E Camelback Road, Ste 250, Phoenix AZ 85018 USA
e...@indirect.eom telephone (602)661-3911 Fax (602) 451-7617

Figure 2.3: The first commercial spam message sent by two lawyers to over 6000 newsgroups to advertise a green card lottery

research company, estimate that spam will cost the US $17 billion in lost productivity in 2005 with the cost to each individual of manually filtering the spam they receive at $718 per year[15]. They also estimate that the cost to UK companies is UK/1.2 billion[16].

Ultimately spam is reducing the value of email. Most users are now aware that their email messages may not arrive or be seen as they may be stopped by spam filters. In a sense, spam is becoming more acceptable, a necessary evil, as general opinion is that there are no totally effective ways of stopping it. There are many different approaches to combatting spam including new legislative measures, authentication techniques and email/spam filtering. The objective of this chapter is to outline the different techniques for coping with spam with particular emphasis on the filtering approaches. We discuss the different machine learning techniques that have been applied to spam filtering and the challenges facing the application of such techniques to spam filtering.

[15] informationweek.com/shared/printableArticle.jhtml?articleID=60403016
[16] www.personneltoday.com/Articles/2005/03/09/28519/Spam+costs+UK+businesses+%C2%A313bn+year.htm

2.1 Anti-Spam Legislative Measures

The *Controlling the Assault of Non-Solicited Pornography and Marketing Act* of 2003, more commonly known as the CAN-SPAM Act, establishes the first national standards for the sending of commercial email in the US. It came into effect on the 1st January 2004. The Act allows email marketers to send unsolicited commercial email as long as it contains all of the following:

- a valid subject line which cannot mislead the recipient about the subject matter or contents of the email,
- valid header and routing information; the *From* and *To* and routing details must be accurate and identify the sender,
- a label if the content is adult,
- an opt-out mechanism which allows the recipient to indicate that they do not wish to receive any more email from the sender. The Act gives the sender 30 days to honour the request and 10 days to stop sending emails to the requestor's email address. It also makes it illegal to sell or pass on the email address.

Other common spamming practices, such as harvesting, dictionary attacks, Internet protocol spoofing[17], hijacking computers through Trojan horses or worms, or using open mail relays for the purpose of sending spam, can make a CAN-SPAM violation an "aggravated offence". The USA's first felony prosecution of spammers found Jeremy Jaynes guilty of using bulk email to market fake products. He was sentenced in April 2005 to 9 years in jail.

There is considerable controversy about the Act. There is a body of opinion that believes that the Act, in effect, legalises the sending of spam[18] because as long as an opt-out mechanism is included in an email it gives bulk advertisers permission to send unsolicited commercial email. The CAN-SPAM Act, as it is a federal law, also overrides state law. Certain states, such as California had more severe anti-spam laws[19] which were overridden by the federal Act. California's anti-spam law took an opt-in approach, where senders had to have the recipient's permission before sending commercial email to

[17] The creation of IP packets with a forged (spoofed) IP address
[18] www.spamhaus.org/news.lasso?article=150
[19] www.pcworld.com/downloads/countit.asp?fid=23113&fileidx=1

a Californian address and fines of up to $1000 could be applied for each message and up to $1,000,000 for each advertisment.

The Act only regulates spam sent from within the US and according to Sophos's survey[20] covering the first quarter of 2005, the US is still the top spam producing country in the world with over 35% of the spam received in Sophos's world-wide spam traps originating in the US. South Korea and China hold second and third place with just under 25% and 10% of spam respectively, originating from these countries. Proponents of CAN-SPAM can argue it is having some effect as figures published by Sophos a year earlier, just after the CAN-SPAM Act came into effect, show the US as the top spam producing country with 56% of spam originating there[21]. However, statistics from MessageLabs[22] (see Figure 2.4) show the volumes of spam email as a share of global email have continured to increase significantly since the Act came into play.

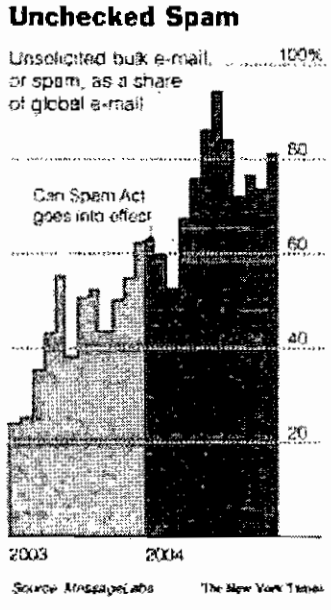


Figure 2.4: Spam as a share of global email (Source: MessageLabs and New York Times)

Although the European Union anti-spam directive[23] also outlaws disguised identities and invalid IP addresses, it takes a contrary approach to the CAN-SPAM Act advocating an opt-in policy. The directive states that email marketing can be used in a situation where a prior business relationship exists or where the receiver has opted-in or agreed to receiving marketing emails. Enforcement of the directive is left to the individual coun-

[20] www.sophos.com/spaminfo/articles/dirtydozen05.html

[21] www.sophos.com/spaminfo/articles/dirtydozen.html

[22] MessageLabs provides messaging security and management services to businesses, see www.messagelabs.com

[23] europa.eu.int/eur-lex/pri/en/oj/dat/2002/l_201/l_20120020731en00370047.pdf

tries. By April 2004, the EU had sent warning letters to eight countries who had not implemented the directive, including Germany, France, Belgium, Netherlands, Greece, Portugal, Luxembourg and Finland but the directive has now been implemented into law by all European Union member states.

An issue with many of the anti-spams laws in European countries is the problem of bringing actions against foreign-based spammers. This is due mainly to the difficulty of conducting an investigation in a foreign country and the problems linked with ensuring the execution of any successful convictions. More recently there have been moves between various countries to address the challenges of information sharing and gathering with respect to spamming. The US, UK, Korea and Australia have entered into various memoranda of understanding on spam to improve co-operation between enforcement agencies on cross-border cases. The London Action Plan[24] encourages the government and public agencies responsible for law enforcement from the participating 27 countries to develop better international spam enforcement co-operation.

The OECD report on anti-spam law enforcement[25] published in May 2005 has concluded that while spam has been identified as a critical and global issue requiring coordinated international action, significant steps are needed for effective national and cross-border enforcement.

2.2 Authentication-Based Anti-Spam Techniques

Spammers frequently exploit the fundamental flaw in email that it is easy to make an email message look like it came from any address. This has made it difficult to blacklist spammer email addresses and has resulted in the latest email scourge of *phishing*, emails which appear to come from financial institutions and request confirmation of account or credit card and PIN numbers. Authentication-based techniques[26] are techniques that attempt to combat this, by requiring the sender to authenticate themselves in some way.

An early authentication technique is the challenge-response system[27]. These systems are installed at the mail server level and only admit emails where the sender is known to the receiver. When an email from an unknown email address is received at the user's mailbox the challenge-response system issues a challenge email indicating that the sender must respond before the email will be delivered to the recipient, as illustrated in Figure 2.5. The

[24] www.ftc.gov/os/2004/10/041012londonactionplan.pdf
[25] www.oecd.org/dataoecd/18/43/34886680.pdf
[26] www.emailauthentication.org/
[27] en.wikipedia.org/wiki/Challenge-response

objective here is that spammers who generally send machine-generated bulk emails would not be in a position to respond. In addition, spammers often use forged or spoofed email addresses and challenge emails would never be received. More sophisticated challenge-response systems attempt to ensure the response email can not be machine-generated. A common way of achieving this is to include in the challenge email a distorted image of some text, known as a *captcha* (von Ahn et al. 2004), and require that the text is typed into the response email.

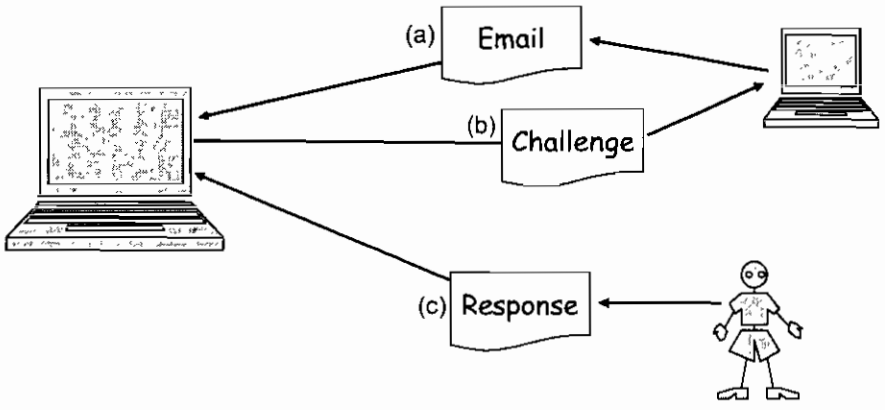


Figure 2.5: A Challenge-Response authentication system: (a) Email is received from an unknown sender (b) System issues a Challenge email (c) Response requires human input

There is mixed opinion as to the effectiveness of the challenge-response systems. Opponents to them complain about the excessive use of bandwidth used by the challenge and reponse emails. In addition, with the increased use of online e-commerce, there is a large group of legitimate emails that will not get through a challenge-response system, such as any emails that are machine-generated by legitimate non-human online interaction, e.g. online shopping, ticket bookings etc. Effective challenge-response systems need to be configured so that they never challenge emails sent by other challenge-reponse systems which would lead to a continuous loop of emails sent back and forth between the challenge-response systems. Also these systems should be able to be configured not to challenge emails from user subscribed mailing lists as these emails will never pass the test. They can also cause problems by issuing challenges to faked email addresses resulting in an innocent party receiving challenge emails from all on a spammer's mailing list (this is known as a Joe-job). There are a number of downloadable challenge-response systems available including TMDA[28], Mailblocks[29] and Qurb[30].

[28] tmda.net
[29] www.mailblocks.com
[30] www.qurb.com

Sender Policy Framework (SPF)[31] is another authentication approach that uses the existing Domain Name Service (DNS) system to authenticate the sender of the email. An organisation or ISP that sends email publishes an SPF record that lists the IP addresses of the servers that it sends email from. This SPF record is an extension to their current DNS entry that lists servers that receive email. When email is received from a mail server the SPF record for that domain is checked to make sure that server is allowed to send email. Those emails that don't pass the check are not accepted, see Figure 2.6.

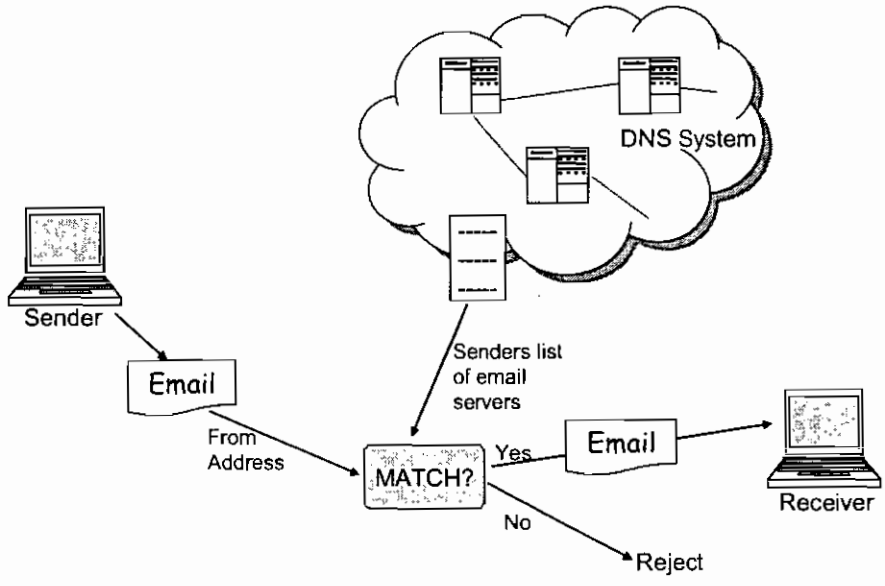


Figure 2.6: Sender Policy Framework

One of the limitations with SPF is that as it only checks the 'From' header field, forwarded mail will not pass the test. Microsoft's Sender ID[32] (which now incorporates SPF) proposes a second standard called Purported Responsible Address (PRA). PRA extracts the purported sender from the email headers and validates that. Both SPF and PRA are being considered at the moment as Internet standards. It seems unlikely that PRA will be chosen as the single standard as Microsoft is attempting to patent the technology and will seek licences from software makers that implement it. This would be against the GNU General Public License which is the most popular license for free software. Open source advocates (e.g. Apache Software Foundation) oppose it for these reasons.

Yahoo's Domain Keys also authenticates the sender using optional extensions to the DNS system. Domain Keys work by adding a digital signature to the email encorporating

[31] www.pobox.com

[32] www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx

a hash signature of the email contents. The owners of the domain names publish a public key with the DNS entry for the domain name which can be used to validate the digital signature and to decrypt the hash signature. By recalculating the hash signature on the email contents and comparing it with the decrypted one the receiver can ensure that email has not been tampered with. The forwarding issue arises with Domain Keys too in that the contents of an email can be legitimately changed when the email is forwarded which will result in the hash signatures not matching. To get around this limitation it is recommended that forwarded mails are re-signed or Domain Keys is used with other authentication techniques.

Authentication techniques such as SPF, SenderID and Domain Keys will not end spam, but do make it possible to identify and track the sources of spam. As long as spammers actively register their SPF or Domain Key records with their domain names, their messages will not be stopped. But these techniques will prevent spoofing and phishing attacks. As they always do, spammers are adapting in order to circumvent these anti-spam measures and are publishing SPF records for their domain names[33]. It is relatively easy for a spammer to buy and register a cheap disposable domain name.

2.3 Spam Filters

There are a number of different approaches to spam filtering and no single filter uses just one approach. The most successful filters apply a number of approaches. These include header analysis to check for spammer specific characteristics in the email, Realtime Block Lists which allow a check to see if the email is from known spammers, collaborative filtering which allows a group to share information about spam and content based filtering which attempts to analyse the content of the email and determine if it is spam. This section outlines each of these potential approaches to combatting spam.

2.3.1 Header Analysis

The objective of header analysis[34] is to check for spammer specific characteristics in the email. These are features the spammer includes to spoof or forge the email including *From:* header fields not including a real name, or fictitious, invalidly or unusually formed *From:* addresses. It can also include invalid routing information including forged *Received:*

[33] www.physorg.com/news1059.html
[34] abuse.msu.edu/email-tracking.html

headers to point to somebody else sending the mail[35]. It normally includes checking dates in the email header as spammers are inclined to forge dates, for instance, to get the email to the top of your inbox.

2.3.2 Realtime Block Lists (RBLs)

Realtime Block Lists (RBLs) are real time lists of the IP addresses of machines that either send or relay spam. Filters can access these lists and check to see if any of the IP addresses in the routing details of the received email are listed. These lists are be accessed directly or provided in periodic datafeeds to many of the Internet ISPs, corporations, universities and such like. Other types of RBL are realtime lists of the IP addresses of illegal third party exploits including open proxies, worms/viruses with built in spam engines and other types of trojan-horse exploits. There are a significant number of RBLs available which an organisation or individual can use including SBL[36], Open Relay Database (ORDB)[37], Trend Micro RBL+ Service[38], Composite Block List CBL[39], Blitzed Open Proxy Monitor BOPM[40], NJABL[41], Exploits Block List XBL[42] and many more. A comprehensive list of RBLs is available[43].

RBLs are a popular and reasonably successful method of blocking email from known spammers but there is a body of opiuion that is against the use of RBLs. An IP address of a non-spammer can end up on one of these lists and it can take a long time to get it removed. Philip Jacob argues the technical and social problems with RBLs concluding that RBL usage hurts small and medium size organizations whose proportional value in the network is small but who can easily be damaged by being listed on an RBL[44].

URL filtering is an associated technique which is based on extracting the linked URLs or domains from an email and checking them against known blacklists such as SURBL[45]. URL blacklists differ from the more well-known IP address blacklists as they allow you to block messages that have spam hosts mentioned in message bodies rather than blocking the actual senders.

[35] email.about.com/cs/spamgeneral/a/spam_headers.htm
[36] www.spamhaus.org
[37] www.ordb.org
[38] www.trendmicro.com/en/products/nrs/rbl
[39] cbl.abuseat.org
[40] opm.blitzed.org
[41] www.njabl.org
[42] www.spamhaus.org
[43] rbls.org/
[44] theory.whirlycott.com/ phil/antispam/rbl-bad/rbl-bad.html
[45] www.surbl.org/

A technique analogous to blacklisting is whitelisting where a list is maintained of the email addresses from which you receive email and if the address of a received email is on the whitelist it is delivered straight away bypassing the filters completely.

2.3.3 Collaborative Filters

Collaborative filters (Gray and Haahr 2004, Damiani et al. 2004) work on the premise that similar spam is sent to lots of people. It is a simple concept, a signature is computed on each email that is received and is compared to a database of known spam. If the signature matches one in the database then the email is classified as spam. The collaboration comes from a group of email users working together to maintain the shared database. If one in the group receives a spam email that is not in the database, it can be added to the database so it is available for all other users in the group. The algorithm used to calculate the signature is key to the effectiveness of the system (Gray and Haahr 2004). Robust or 'fuzzy' algorithms which ignore slight changes in the text are more suitable, e.g. two emails identical except for the salutation at the start should not generate different signatures. Fuzzy hashes have to be more "content-aware" to be able to ignore differences in the irrelevant content of the email while not ignoring differences in the relevant content. There are a number of sucessful collaborative filters available today such as Vipul's Razor[46], Distributed Checksum Clearinghouses (DCC)[47] and Cloudmark Desktop[48].

2.3.4 Content-Based Filters

Most email users can identify a spam email in seconds by simply looking at it. Perhaps it is due to the content of the subject text itself or the fact that there are a lot of uppercase characters or dubious punctuation used. Content-based filters attempt to use the textual content of the email to determine whether the email is spam or not. They divide into two types, rule-based filters and filters that use machine learning techniques.

Rule-based filters use a series of rules about the actual words or phrases included in the subject or body of the email such as the existence of the word "Viagra" or offering something "Free". Rules about the structure of the text are often included too, such as the proportion of HTML markup in the text, or the fact that the font size in the HTML is very small. Each rule can have a score and if the accumulation of the scores for all the

[46] razor.sourceforge.net
[47] www.rhyolite.com/anti-spam/dcc
[48] www.cloudmark.com

rules that were fired is greater than a certain threshold then the email will be classified as spam.

The limitation of rule-based filtering is that it is a knowledge intensive and time-consuming process to review the spam emails to determine the rules. As spammers change their emails to bypass the current rules (e.g. including obfuscation in key words like 'F.R.E.E') new rules have to be added.

The objective of using machine learning in spam filtering is to build a system that automatically *learns* from examples of known spam and legitimate emails and uses them to categorise new email as spam or legitimate. Such filters use a representative collection of legitimate and spam emails to identify characteristics of email that are predictive of spam and legitimate email. Machine Learning and its application to spam filtering is discussed in detail in the next section.

The fact that the machine learning system automatically *learns* how to identify spam email from a collection of emails is an advantage over the rule-based approach. The time consuming knowledge acquisition process of identifying all the rules of spam email, which is the limitation of the rule based approach, is not required in machine learning. The machine learner uses the examples of spam and legitimate email that it is given as training data to *learn* the characteristics of spam.

For an online learning task such as spam filtering the machine learning system needs to be updated to take into account new types of spam not used in the initial training. This is a challenge facing any spam filter and is discussed in the Section 2.4.1. This continual learning needs to be directed by the user who can determine whether the classification performed by the system is correct or not.

There is no single technique that is 100% successful at identifying spam email. Most filters that are available commercially or as open source use a combination of techniques. SpamAssassin[49] which is the most popular open source filter available[50], uses all of the techniques discussed above including header analysis, RBLs, rule-based filtering, Bayesian filtering and collaborative filtering.

[49] spamassassin.apache.org/

[50] SpamAssassin was voted the top open source anti-spam product in Datamation's Product of the Year 2005, see itmanagement.earthweb.com/secu/article.php/3481971

2.4 Machine Learning and Spam Filtering

Machine learning (Simon 1983, Mitchell 1997) is the ability of a computer system to improve its current performance based on its past performance. Machine learning systems use a representative collection of previous problems, with their solutions, to identify characteristics of the problem that they are trying to solve. This training data is used to construct a model that best generalises to all possible examples of the problem, see Figure 2.7. This model is then used to classify new, unseen problems.

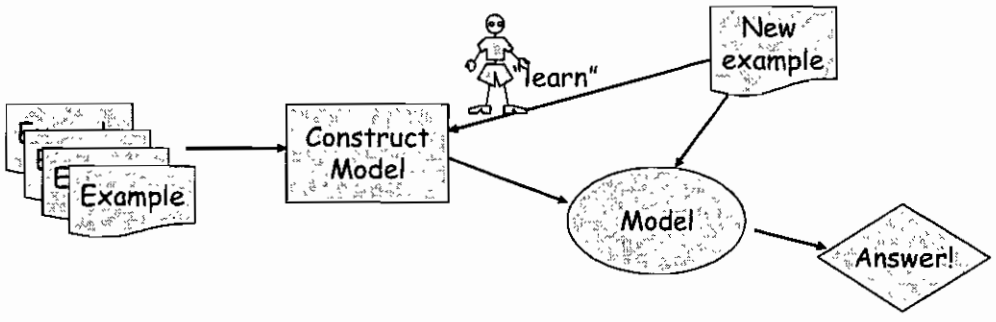


Figure 2.7: The Machine Learning process

In more recent times, a significant amount of research has applied the techniques of machine learning to spam filtering. Spam filtering is a classification problem; given a new email the system has to assign a label to it, i.e. *classify* it as a spam or a legitimate email. A spam filter is therefore a type of classifier.

Spam filtering in general, and specifically the application of machine learning to spam filtering, presents a number of challenges. This section will discuss these challenges and reviews existing research in this area.

2.4.1 Challenges of spam filtering for machine learning

No machine learning classification technique is likely to be perfect. Classification errors are inevitable but the extra challenge facing any spam filtering technique is the cost of misclassifying legitimate emails as spam. Misclassified legitimate messages are known as False Positives (FPs) and are unacceptable to most email users. Fawcett (2003) classifies this problem as "unequal and uncertain error costs". It is difficult to quantify the cost of false positives. Missing a legitimate email to one person may be just an inconvenience whereas to another it could mean the loss of a business opportunity. One global entertainment conglomerate estimated that a single lost email from an important customer could cost them more than $100,000 in lost business while a US state legislator said that an

incorrectly blocked mail from a constituent could cost the votes of the constituent, his family, friends and neighbours[51]. In spite of the likelihood of false positives being low with most spam filters, without a solid guarantee many users of email are negating the work done by the filters by insisting on seeing all emails before they are deleted. Ferris Research published figures in 2003 that estimated the cost of FPs to US businesses of $3.5 billion[52].

A second significant challenge of spam filtering is concept drift; concept drift refers to the fact that the very characteristics which may be predictive of spam are constantly changing over time. Concept drift is very evident in spam, as new opportunities are constantly being exploited by the spammers. In April 2005, following the death of Pope John Paul II, a surge of religious spam emails was identified by Email Systems[53], an email management and filtering company, who found that one in ten of the spam emails they filtered was related to religion[54]. This religious spam included offers of free biographies and audiobooks related to the deceased pope and requested donations towards a cathedral in his honour. Seasonal effects are also noticeable in spam with, for example, an increase in spam advertising weight loss products after Christmas[55].

One factor that drives the significant concept drift in spam is the virtual arms race that exists between the spammers and the filter producers. The spammers continually change the content and structure of the spam emails as the filters are modified to catch them. If a content filter uses the word *Viagra* to identify a spam email, by changing it to *Vi@gra* or *V I A G R A* the email may bypass the filter, see an example of this in Figure 2.8.

For collaborative filters, random text can be added to the end of an email message (known as *spam salad*[56]) to make each spam message different, see Figure 2.9. This will result in a different signature being generated for each message, leaving little basis for similarity between messages, which is the cornerstone of collaboration.

Concept drift is also evident in the structure of the messages that spammers send. Spammers have changed from using the original text-based messages such as Canter and Siegel's in Figure 2.3 to using HTML, see Figure 2.10(a). Once the filters adapted to identifying HTML spam, the spam was obfuscated by including redundant HTML tags

[51] whitepaper.informationweek.com/cmpinformationweek/search/viewabstract/70122/index.jsp
[52] www.enterpriseitplanet.com/security/news/article.php/2246371
[53] www.emailsystems.com
[54] news.ft.com/cms/s/3083f910-baa7-11d9-a27b-00000e2511c8.html
[55] www.polesoft.com/av126_News_News_news_8.html
[56] www.bloggerforum.com/blog/2004/02/spam-salad.html

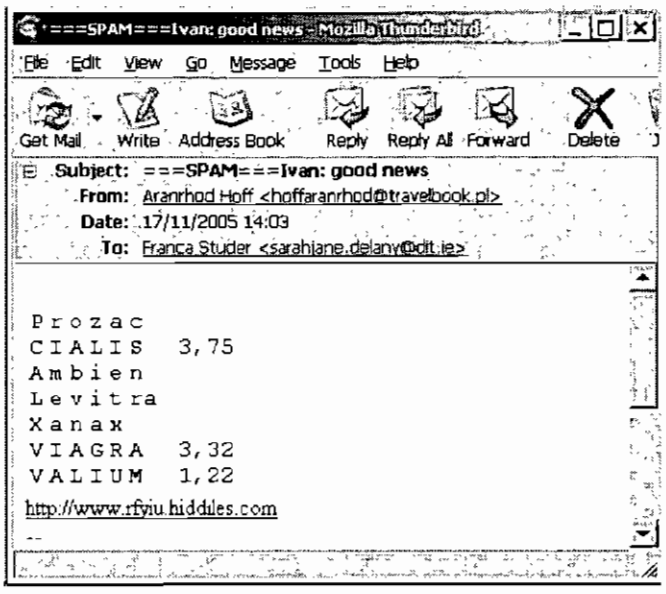


Figure 2.8: Spam email that obfuscates potential key words that may be searched for by a filter, by separating each character with a space

in the source, invisible when the message was displayed but confusing the filter's pattern matching routines, see Figure 2.10(b).

Spammers more recently have moved back to simple text based messages that resemble closely the structure and content of a legitimate message from a friend or colleague, see the example in Figure 2.11. Another trick used by spammers to circumvent the filters is to embed the message in an image, so there is little 'content' for the filters to work with, see Figure 2.12.

Another problem with identifying spam is that what is spam to one person may not be considered spam by someone else. Consider the email shown in Figure 2.13, it was sent by a student to all the staff of a large Computer Science department using a mailing list. Most people who received it would consider it spam, unsolicited bulk email, but it is not unreasonable to assume that it might have been of interest to one or two people within the department. This indicates that there is a need for a personal spam filter, one that can identify spam based on what the individual operating it considers to be spam.

Applying machine learning techniques to spam filtering presents its own set of challenges. A machine learner acquires or learns a general concept from specific training examples; it uses available examples of data to build a model that best generalises to all possible examples. One of the issues facing a machine learning spam filter is which examples of spam and legitimate email should be used as training data; most email users receive tens or even hundreds of emails a week. Associated with this is the problem of

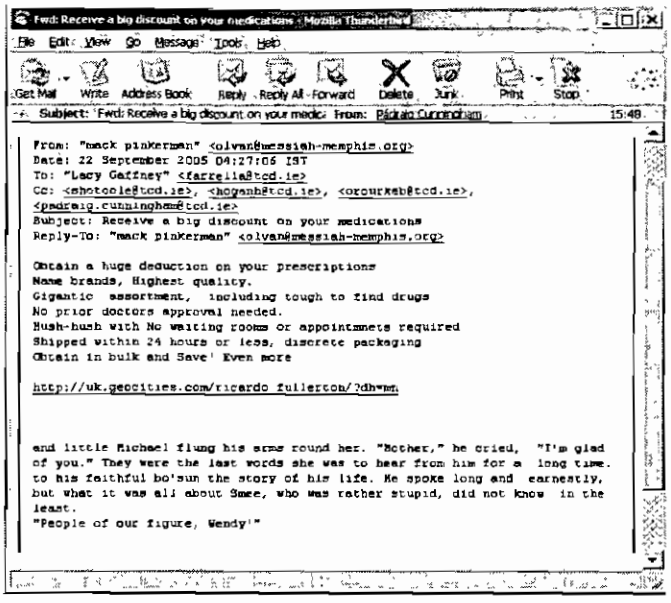


Figure 2.9: Spam salad, a spam email including random text to confuse filters

a skewed and drifting class distribution. The proportion of spam to legitimate email is different for each person. The amount of spam received by an individual can depend on the email address, the amount of exposure it has had on the Internet and the extent of filtering performed. Similarly the amount of legitimate mail received varies from person to person.

A key task in a machine learning system involves identifying the features or characteristics of the domain in question that are most predictive of what the system is attempting to do. Spam filtering is a text categorisation problem. With text categorisation problems the characteristics or features upon which the learning model is built are normally the textual content, the words, letters or phrases in the text. Parsing a training set of sample emails can result in thousands of words or features. There is an exponential increase in the complexity of systems as the number of features increase which is known as the curse of dimensionality (Bellman 1961). In addition, certain machine learning techniques such as neural networks or decision trees are not amenable to domains with high dimensional data.

Finally, spam email is a diverse concept for a machine learner to learn. Spam advertising viagra has little in common with spam offering free mortgages or investment opportunities. Certain machine learning techniques will find it difficult to build a single model that can distinguish between financial, pharmaceutical, pornographic and religious spam.

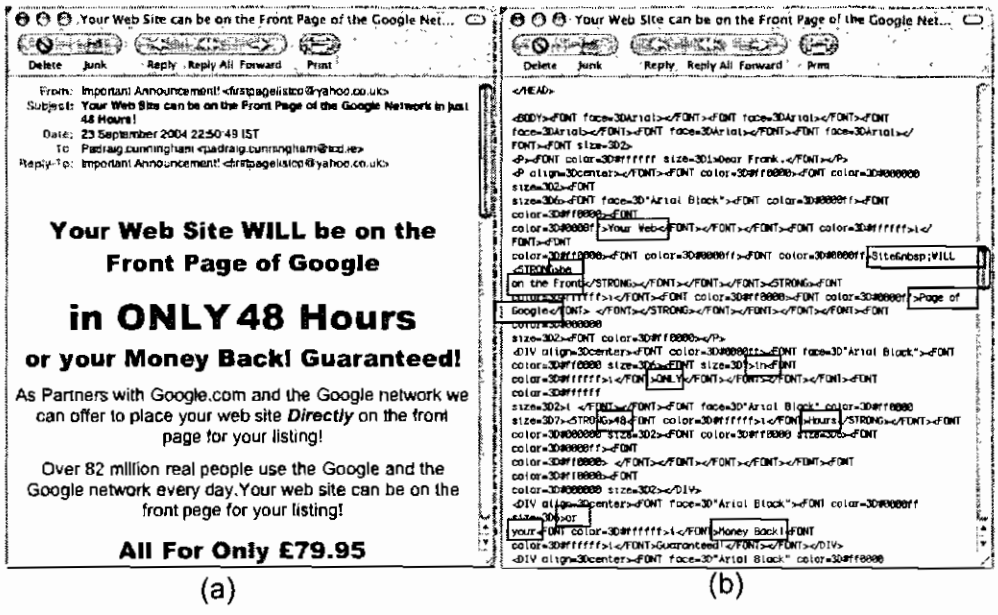


Figure 2.10: (a) Spam email that uses HTML (b) The corresponding source that illustrates the superfluous and redundant tags included to confuse spam filters

There have been a number of machine learning techniques applied to spam filtering. The following sections outline the state of the art in spam filtering research discussing the classifiers used, the feature representation and pre-processing performed, if any. It also discusses the issue of cost, what costs have been applied to FPs and cost measures used by the different rescarchers.

2.4.2 Machine Learning Techniques used in Spam Filtering

Research into the machine learning techniques used for building spam classifiers fall into two categories, those that are evaluated on static datasets in offline environments and those that are evaluated in online, real-time environments. The majority of the research falls into the former category with little published research showing how effective these techniques are at actually filtering real email. This section will discuss this research under both these categories.

Offline Evaluations of Machine Learning Techniques for Spam Filtering

The two most common approaches to spam filtering are Naïve Bayes (Mitchell 1997) and Support Vector Machines (Vapnik 1999). A considerable number of evaluations using these approaches have been published. This section will discuss the effectiveness of the different techniques used under the heading of the classification technique used.

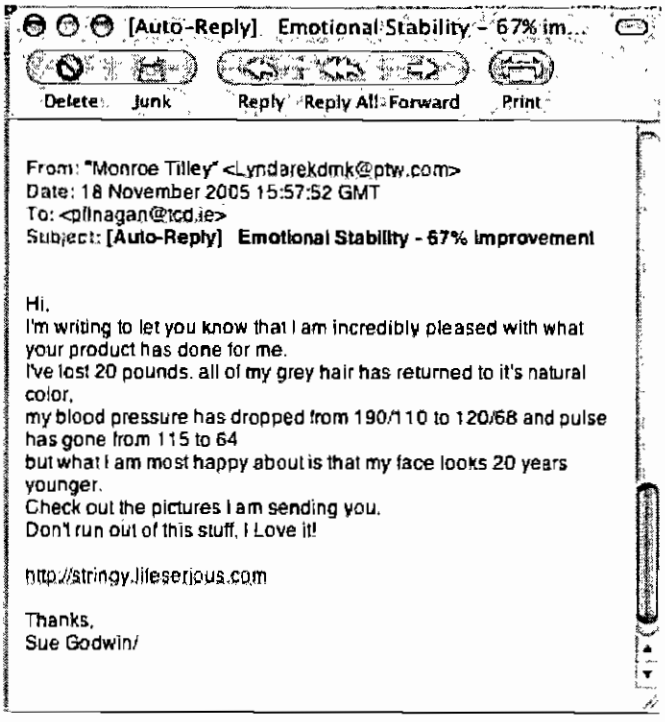


Figure 2.11: Spam email that resembles closely a legitimate email

Naïve Bayes The most common commercial machine learning approach to spam filtering is the use of Naïve Bayes classifiers (Androutsopoulos et al. 2000a, Sahami et al. 1998, Pantel and Lin 1998). Naïve Bayes is a probabilistic classifier and is discussed in detail in Section 3.4.1. Briefly it calculates and uses the probablity of certain words/phrases occurring in the known examples to categorise new examples. Naïve Bayes has been shown to be very successful at categorising text documents (Lewis and Ringuette 1994, Lewis 1998). A number of commercial spam filtering products and systems claim to use such learning including SpamTrapper[57], GFI Mail Essentials[58], Outlook Spam Filter[59], G-Lock SpamCombat[60], MailFrontier[61], SpamSleuth[62] and many more. There are also a number of freeware spam filter implementations available for download and installation including Bayesian Mail Filter[63], Bogofilter[64], SpamAssassin[65], SpamBayes[66], PopFile[67],

[57] brigsoft.com/outlook-spam-filter/
[58] www.gfi.com/mes/
[59] www.outlook-spam-filter.com/
[60] www.glocksoft.com/sc/
[61] www.mailfrontier.com/
[62] www.bluesquirrel.com/index.html
[63] sourceforge.net/projects/bmf
[64] bogofilter.sourceforge.net
[65] spamassassin.apache.org
[66] spambayes.sourceforge.net/
[67] popfile.sourceforge.net/

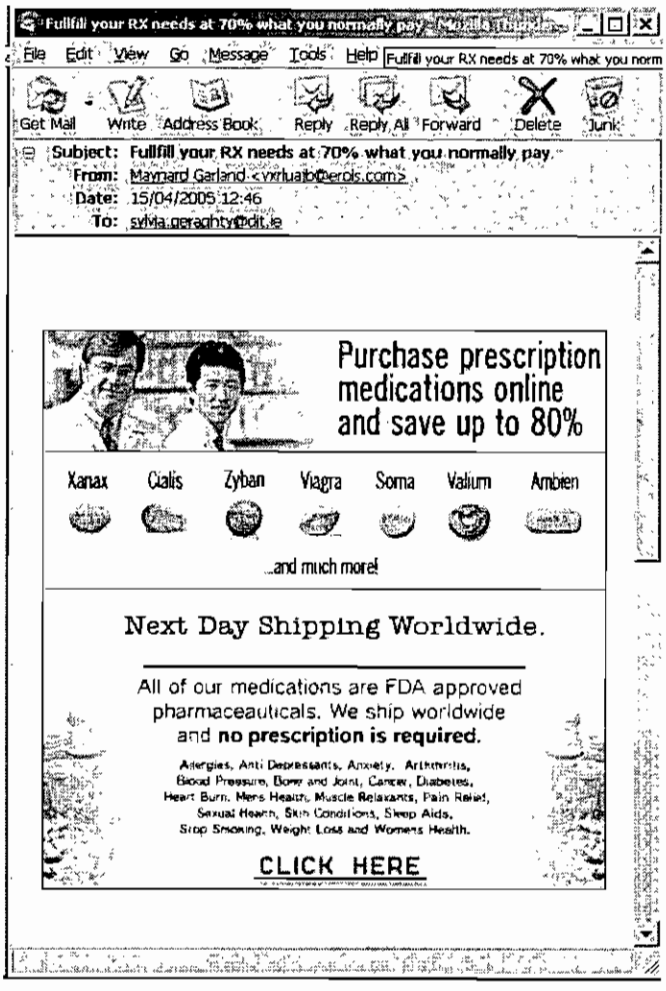


Figure 2.12: Spam email that uses an image to bypass the filters

SpamTUNNEL[68] and Python Anti-Spam Proxy (PASP)[69] with various comparisons available available on the web[70].

Naïve Bayes has been applied to the more general process of analysing and organising email messages and has been shown to perform better than rule-based techniques (Provost 1999, Rennie 2000). Early research in the more specific task of spam filtering includes Sahami et al. (1998) and Pantel and Lin (1998). Sahami et al. achieved a 5.4% error rate with .02% false positive rate classifying 222 emails whereas Pantel and Lin achieved appoximately 4% error rate with just over 1% false positives classifying 600 or so test emails. This early work used very small training and test sets and did not consider any of the issues associated with the concept drift in email.

[68] uiorean.cluj.astral.ro/

[69] sourceforge.net/projects/pasp

[70] URLs home.dataparty.no/kristian/reviews/bayesian/ and freshmeat.net/articles/view/964/ offer comparisons of some of the software applications listed above

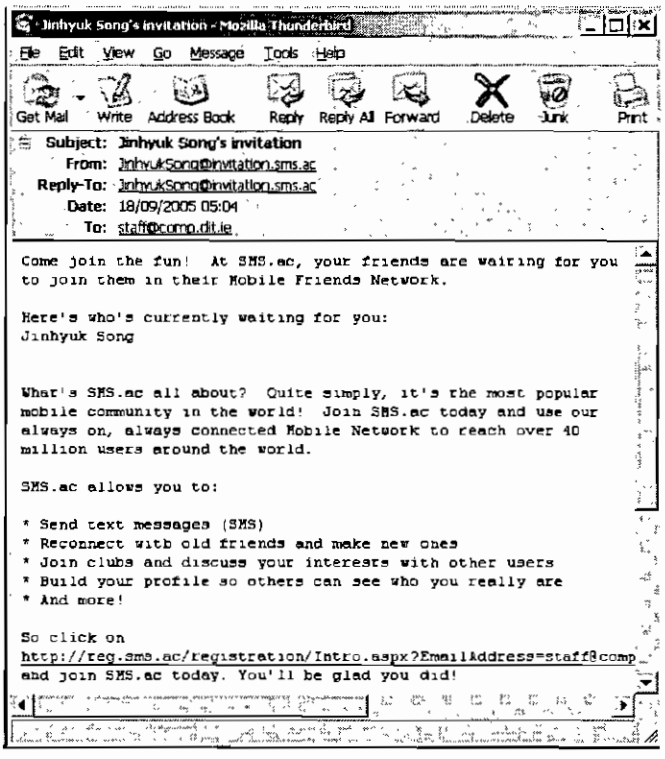


Figure 2.13: Spam email that may be considered spam to one person and legitimate to someone else

Androutsopoulos et al. (2000a) performs a more comprehensive evaluation of Naïve Bayes and concludes that it does perform well in spam classification tasks but they point out that additional safety nets are needed to ensure email identified as spam is not deleted automatically but is available to the user in some way. Androutsopoulos et al. (2000b) extended their work and compared Naïve Bayes to a rule-based filter using keyword patterns (part of Microsoft Outlook 2000). They concluded that the Naïve Bayesian filter was far superior to the rule-based filter. However their experiments were conducted on a static dataset of 1099 messages and the rule-based filter included just 58 rules. On the other hand Hidalgo et al. (2000)'s comparison of Naïve Bayes with C4.5 decision trees and PART[71] (Frank and Witten 1998) found the decision tree learners both outperformed Naïve Bayes.

Androutsopoulos et al. (2000c) compared the Naïve Bayes approach to a memory based approach and concluded that the two learning methods performed equally well and both outperformed the mail reader Outlook 2000's keyword-based rule-based filter while Cunningham et al. (2003)'s preliminary research into applying case-based reasoning to spam filtering found that CBR outperformed Naïve Bayes. Androutsoponlos and his

[71] PART is a rule learner where the rules are derived from partial decision trees.

colleagues extended their research to perform a extensive empirical evaluation of using memory-based learning (using a k nearest neighbour classifier) for spam filtering of mailing lists (Androutsopoulos et al. 2000c, Sakkis et al. 2003). Their results indicate that anti-spam memory-based learning is practically feasible especially combined with the additional safety nets also discussed in Androutsopoulos et al. (2000a). More interesting is their exploration of the various parameters such as feature weighting, distance weighting and neighbourhood size. They concluded that feature weighting and distance weighting is beneficial but neighbourhood size is inconclusive and varies in an unintelligent way with the number of features and the cost model used.

O'Brien and Vogel (2003) compared Naïve Bayes with the Chi-squared test, a statistical test, proposed by Kilgarriff and Salkie (1996), that measures similarity and homogeneity in text corpuses. Their evaluation showed that both techniques performed well. The authors compiled their own dataset of spam and legitimate email on which to perform their evaluation. Although they used a respectable 759 emails for training, their evaluation was based on merely 67 test emails which suggests that a more comprehensive evaluation would be required to confirm their results.

Chuan et al. (2005) compared Naïve Bayes to two neural network approachs and found that the neural network approaches both performed better than Naïve Bayes. They suggest that this is because of the independence assumption of Naïve Bayes, the neural network can take into account the relationship between words in the text. Chuan et al. use a subset of 1000 emails selected randomly from a larger dataset. They use only the top 100 features (chosen using Mutual Information) from the available feature set. They indicate that the training time increases dramatically using any higher numbers of features. This confirms the unsuitablility of using neural nets in a more real time scenario.

Support Vector Machines The success of using Support Vector Machines (SVM) at classifying text documents (Joachims 1998, Dumais et al. 1998, Cardoso-Cachopo and Oliveira 2003) has prompted significant research into applying them to spam filtering (Drucker et al. 1999, Kolcz and Alspector 2001). SVMs are kernel methods whose central idea is to embed the data representing the text documents into a vector space where linear algebra and geometry can be performed (Cristiani and Scholkopf 2002). SVMs attempt to construct a linear separation between two classes in this vector space. Details of the SVM technique are included in Section 3.4.2.

Drucker et al. (1999) compared SVMs to Ripper, a rule-based induction algorithm (Cohen 1995), and a boosted ensemble of C4.5 trees (Quinlan 1997). Both the SVM and the boosted trees performed well but the authors concluded that the SVM was superior as it was significantly faster to train. They also note that an advantage of the SVM is that it can handle large feature sets avoiding the feature selection process. Zhang et al. (2004) also found that SVMs were the top performers (along with AdaBoost (Schapire 2002) and maximum entropy model (Berger et al. 1996)) in their evaluation. Their conclusions were that SVMs were not sensitive to feature selection and easily scalable to high feature dimension with good performance across a number of datasets. A disadvantage of using SVMs for an online, real time task such as spam filtering is due to the concept drift in spam. As the concept that the learner is modelling is constantly changing the model needs to be updated with new examples of spam and legitimate email to keep the performance up. This is a time-consuming task for an SVM and an open research question, one that has provoked much research into the area of updatable SVMs (Syed et al. 1999, Klinkenberg and Joachims 2000, Rüping 2001).

Ensembles There has been some research also in applying ensembles of classifiers to the problem. An ensemble of classifiers combines the results from a number of base classifiers where each base classifier is constructed on a snbset of the training examples. Sakkis et al. (2001) combined a k-NN and a Naïve Bayes classifier in a two-member ensemble and found that the ensemble performed better than either of its constituent classifiers. Carreras and Márquez (2001) found that an ensemble of boosted decision trees performed better than a single decision tree classifier and a Naïve Bayes classifier although Drucker et al. (1999)'s results found that an SVM outperformed an ensemble of boosted decision trees.

Other techniques There has also been research in spam filtering using more unusual and interesting techniques. Gee (2003) applied the statistical technique of Latent Semantic Indexing (LSI) (Landauer et al. 1998) to the problem. The technique of LSI derives a statistical correlation between all the terms and documents in a corpus of text documents to try to overcome the problems with direct lexical matching. It attempts to capture semantic similarities between words and phrases and these can be used to compute similarities between documents that do not contain the same words or phrases. Gee found that LSI was a viable method for classifyiug spam with similar results to those achieved using Naïve Bayes, although LSI did consistently misclassify certain legitimate emails. The

obvious drawback to using LSI is similar to that of SVM in that handling concept drift poses a challenge. Adding new documents requires completely rebuilding the semantic relationships between all documents.

Oda and White investigated the use of an artificial immune system to protect against spam (Oda and White 2003a,b). The human immune system works when antibodies, the receptors on the good white blood cells bind with the pathogen (the harmful virus or bacteria) and destroy it. Antigens are the features used to identify the pathogen. The antigens in the artificial spam filtering immune system are the email contents including headers and text, and the antibodies are implemented using regular expressions, patterns which can match a variety of strings. Oda and White found that the results from the arificial immune system were comparable to results from SpamAssassin, a large primarily rule-based system[72] on a spam corpus made available by SpamAssassin.

Boykin and Roychowdhury (2005) propose an automated graph theory method to create a social network from a user's email. Using the structural properties of social networks, messages are identified as spam, legitimate or unknown based on clustering thresholds. Their method is able to classify about 50% of a users email as spam or legitimate, leaving the rest to be filtered by other techniques.

Online evaluations of machine learning techniques for spam filtering

There are two key machine learning filtering systems that have been evaluated against live email; Filtron (Michelakis et al. 2004) and Spamato (Albrecht et al. 2005). Filtron is a prototype anti-spam filter that was designed and developed on the results of a comprehensive empirical evaluation of four learning algorithms, Naïve Bayes, Flexible Bayes, LogitBoost and SVMs (Androutsopoulos et al. 2000d). It is a Java implementation that runs on Unix platforms only and was evaluated by a single user over seven months. The classifier used was an SVM as that showed the best performance in the offline evaluation, although the system is configurable and different classifiers can be selected. The system was initially trained on 2313 legitimate emails received by the user and 1826 general spam messages and used 520 features. It was never retrained over the period it was used and the performance was very good with 52 FPs reported out of 5109 legitimate mails received (approx 1.0%) and 173 out of 1623 spam received missed by the filter (10.6%).

Albrecht et al.'s Spamato filter is an open extendable spam filter framework implemented in Java utilising a plug-in filter architecture. The author's initial beta-test eval-

[72] spamassassin.apache.org/

uation used a variety of plug-in filters including a number of URL-based filters, a Naïve Bayes classifier, a rule-based filter and a collaborative filter. This evaluation which filtered approximately 90,000 messages from a dozen users, resulted in 0.5% FPs and 7% spam emails that were missed by the filter.

2.4.3 Preprocessing and Feature Representation

Spam classifiers operate by looking for patterns among the features of spam and legitimate messages and these features are usually related to the content of the message, typically the words themselves. This leads to high dimension feature spaces and so most of the research performs pre-processing on the textual content of the message to reduce the dimensionality of the feature space. Typical operations that are performed are stop-word removal (also known as function removal (Sebastiani 2002)) which is the removal of topic-independent terms such as articles, prepositions, conjunctions etc., and stemming (Frakes and Baeza-Yates 1992) which involves grouping words with the same morphological root, e.g. 'computing' and 'computer'. There is great variety in the pre-processing that is applied in the spam filtering research discussed in Section 2.4.2. The only empirical evaluation of whether pre-processing of the text was beneficial to spam filtering concluded that using a stop list did not seem to have any noticable effect and although there was a slight signs of improvement when stemming was applied, the differences were not statistical significant (Androutsopoulos et al. 2000b). Ahmed and Mithun (2004) devised a rule-based word stemming technique that matched both words that sound alike and look alike which was designed to group words like 'Viagra', 'V.i.a.g.r.a' and 'Vi@gra'. Some authors remove the stop words (Rennie 2000, Provost 1999), others include stemming (Schneider 2003, Androutsopoulos et al. 2000c, Sakkis et al. 2003) while some do neither (Drucker et al. 1999, Kolcz and Alspector 2001, Zhang et al. 2004, Carreras and Márquez 2001). Others attempt dimensionality reduction by removing words occuring in less than a certain number of messages (Sakkis et al. 2003, Michelakis et al. 2004, Hovold 2005). Most convert all text to a single case, although Kolcz and Alspector (2001) keep two copies of any words which are totally in uppercase characters, one copy in lowercase and the second in uppercase.

Documents in text categorisation tasks are typically represented as a vector of terms $d_i = \langle t_1, t_2, \ldots, t_n \rangle$ where n is the number of terms used. The terms are identified by tokenisation of the text in the document. Most of the research used word level tokenisation of the email, parsing on white space. However, O'Brien and Vogel (2003) found the char-

acter level tokenisation proved more effective than word level which is a promising result as the general standard in text classification is word-level tokenisation. Androutsopoulos et al. (2000d) found that using n-grams with $n > 1$ did not improve the efficieucy of a filter over unigrams but Hovold (2005) found that they did improve performance for Naïve Bayes classifiers that use word-position in the calculation of the probabilities.

There is also variation in the part of the actual email that is included in the tokenisation of the text. Most authors include the Subject header field in the tokenisation of the text although others indicate that using headers such as *From:* and *To:* fields can also be useful (Rennie 2000, Drucker et al. 1999). Rennie terms this 'header trimming'. Hovold (2005) fonnd that performance decreased when all headers were included. He concludes that mail headers and HTML should not be included by brute force but a selective inclusion process would be more appropriate. Contrary to this, Zhang et al. (2004)'s evaluation of four datasets indicate that message headers can be reliable and powerful discriminative feature for spam filtering when all header information is included.

The vector representation of the document can either be binary where $t_j \in \{0, 1\}$ or it can be numeric, real-valued i.e. $0 \leq t_j \leq 1$. A binary representation simply reflects the existence of the term in the document. Numeric representation normally represents the frequency of the term in the document. It can be calculated statistically, which is the most common option, or probabilistically. Binary representation of terms is most common among spam filtering research (Sahami et al. 1998, Pantel and Lin 1998, Kolcz and Alspector 2001, Zhang et al. 2004). Androutsopoulos and his colleagues initially used binary representation for their evaluations (Androutsopoulos et al. 2000b,c,a, Sakkis et al. 2001) but in more recent times have switched to a numeric representation (Androutsopoulos et al. 2000d, Michelakis et al. 2004) using the normalised frequency of the term in the document as the value of t_j although an evaluation of numeric versus binary does not seem to have been performed. Drucker et al. (1999) found that binary representation performed better than numeric for SVMs. While Schneider (2003) compared two event models for Naïve Bayes anti-spam filtering and concluded that using a numeric representation performed slightly better than a binary one.

2.4.4 Spam Benchmark Datasets

Androutsopoulos and his colleagues (Androutsopoulos et al. 2000a,c, Sakkis et al. 2001, 2003) collected and used the Ling-Spam corpus[73] of mailing list emails for their evaluations

[73] www.aueb.gr/users/ion/data/lingspam_public.tar.gz

of Naïve Bayes and k-nearest neighbour. This corpus was also used by other researchers (Hidalgo 2002, Gee 2003, Schneider 2003, Zhang et al. 2004). An advantage of using a corpus like this is that it permits comparison across evaluations and can contribute to standard benchmarks. However, this corpus of 2893 messages is a restrictive data set incorporating legitimate email messages sent to a linguistics mailing list and, as it was compiled in 2000, it contains "old-fashioned" spam emails that contain few of the obfuscations common in spam email today.

The PU1 dataset[74] used by (Androutsopoulos et al. 2000d,b, Carreras and Márquez 2001, Zhang et al. 2004) is a collection of personal emails sent to a single individual. It is a smaller dataset of 1099 emails, 481 spam and 618 legitimate. Although as it is personal email it is closer to a real-usage scenario. It is restricted in that it is available only in a encoded format for privacy reasons so the available feature set is fixed.

The UCI Spambase dataset[75] was used by Hidalgo et al. (2000) and does not appear to be available on the Web any more. This dataset consists of 4601 emails, 1813 spam and 2788 legitimate in vector format. The vector contain values for 57 pre-selected features, 47 of these are words and the remaining 10 include specific heuristics, such as the proportion of certain capital letters. This dataset is very restrictive due to the pre-selection of features.

The SpamAssassin corpus[76] has also been used in some research (Zhang et al. 2004, Chuan et al. 2005). It contains 1897 spam and 4150 legitimate message collected from public fora or donated by individual users. The corpus is available in raw message format and was originally compiled in October 2002. It is a useful corpus for server or gateway level filtering but is not representative of an single individual's email.

A number of other researchers collected and used their own datasets for the evaluations of their research (Sahami et al. 1998, Pantel and Lin 1998, Drucker et al. 1999, Kolcz and Alspector 2001, Provost 1999) however, none of the datasets used capture the dynamic (concept drift) nature of spam.

2.4.5 The Cost of False Positives

One of the issues that arises in most of the spam filtering research is the cost of the false positives. A number of researchers attempt to model the costs of misclassification in a spam filter. It is widely recognised among the authors that there are no standard or benchmark costs for the misclassification of legitimate email but a number of cost-sensitive

[74] www.aueb.gr/users/ion/data/PU123ACorpora.tar.gz
[75] www.ics.uci.edu/ mlearn/databases/spambase/
[76] spamassassin.apache.org/publiccorpus/

approaches have been adopted.

Let a True Positive (TP) denote a spam email correctly classified, a True Negative (TN) denote a legitimate email correctly classified, False Positive (FP) denote a legitimate email incorrectly classified as spam and a False Negative (FN) denote a spam email missed by the filter (i.e. incorrectly classified as legitimate), then the accuracy of a filter (Acc) can be calculated as

$$Acc = \frac{\#TPs + \#TNs}{\#TP + \#TNs + \#FPs + \#FNs}$$

and the error of a filter (Err) can be calculated as $Err = 1 - Acc$. Androutsopoulos et al. (2000b) introduces the concept of weighted accuracy and weighted error measure where the cost of a FP is λ times more expensive that the cost of a FN. This is implemented by treating each legitimate message as, in effect, λ messages, weighting it by λ. The weighted accuracy rate (WAcc) is then:

$$WAcc = \frac{\#TPs + \lambda\#TNs}{\lambda(\#TNs + \#FPs) + \#TPs + \#FNs}$$

They propose three cost scenarios; firstly, $\lambda = 1$ where the filter merely flags the spam and lets it through to the user. At the other extreme, $\lambda = 999$ is used when spam messages are deleted automatically and lastly $\lambda = 9$ represents the scenario where spam messages are blocked and some cost is required to unblock and access the message.

Androutsopoulos et al. (2000b) also propose a total cost ratio measure (TCR) because the values of Acc and Err or their weighted equivalents can be misleadingly high. The TCR is calculated as the ratio of the weighted error achieved when using no filter to the weighted error achieved using the filter:

$$TCR = \frac{WErr^b}{WErr}$$

$WErr^b$ is called the baseline filter and is calculated as:

$$WErr^b = \frac{\#TPs + \#FNs}{\lambda(\#TNs + \#FPs) + \#TPs + \#FNs}$$

$WErr^b$ is effectively the actual number of spam emails divided by the total number of emails with the weighting applied to the legitimate emails. The TCR captures to what extent having a filter is better than not having a filter. The larger the value of TCR the better the performance of the classifier. Hidalgo et al. (2000) adopt the weighted accuracy measures as a performance metric but in his further work Hidalgo uses instead ROC and ROCCH analysis for comparing classifiers (Hidalgo 2002). He incorporates difference costing mechanisms into various classifiers and compares performance plotting

their performance on ROC curves. The advantage of this is that the best classifiers for certain cost ratios ranges are identified. Zhang et al. (2004) also use the TCR cost measure to compare performance of different classfiers.

Although TCR is useful for comparing different filters or different classifiers; the difficulty is that the true accuracy and false positive rate is not evident/transparent. In addition, the TCR measure is not useful unless costs have been applied. Consider, for instance, two filters with equal error rates. If filter A is more inclined to misclassify spam and filter B to misclassify legitimate emails, then filter B will have a very high FP rate compared with that of filter A. If no costs are applied in the calculation of the TCR, both will have the same TCR even though filter A is obviously a better performer. So, TCR figures for $\lambda = 1$ are not useful. The main problem with the TCR is that the costs of a FP are unknown and are likely to vary in different circumstances. There is no evidence that a FP is 9 times or 999 times more expensive that a FN.

Kolcz and Alspector (2001) propose a category-specific cost model where the cost of misclassifying legitimate mail is content dependent and the cost of misclassifying spam is uniform. They propose costs of misclassifying legimate mail in the range of 25 for email categorised as promotional offers up to 1000 for email that is categorised as personal. They consider the loss of business email as lower than personal email and assign it a cost of 500. They propose different methods of training the cost-sensitive filter using an SVM as the base classifier. All the methods they investigate for making SVMs cost sensitive performed well and offered clear improvements over the standard SVM classifier which has equal misclassification costs for each class. However, the authors also conclude that the standard SVM which performs well can be turned into a "surpassingly effective" cost-sensitive classifier by adjusting the classification threshold appropriately. This may be preferable if the costs or distribution settings were likely to change which, considering the concept drift that spam is subject to, is highly likely. Kolcz and Alspector also use the TCR to compare the different cost scenarios.

2.5 Conclusions

In this chapter we have outlined the problem of spam and discussed the ways of combatting it. Although in the last couple of years there have been significant laws implemented worldwide to combat spam, the problem seems to be growing. Authentication techniques are becoming popular at preventing the spoofing and phishing that spammers so frequently

do. Even though spammers are adapting to these techniques, they will still help with identifying the sources of spam allowing other anti-spam approaches to block the spam.

There are a number of approaches to filtering spam including header analysis, RBLs, content and collaborative filtering. No single filter uses just one approach as the challenges facing spam filtering are great. The significance of falsely classifying a legitimate email as spam, the fact that spammers are constantly chauging their messages in order to bypass filters and the difficulty of quantifying the cost of getting it wrong all contribute to making spam filtering a difficult problem.

There has been significant research into spam filtering, with Naïve Bayes and Support Vector Machines the techniques that seem to perform best. All the research performed to date has been on static datasets with no effort at tracking the concept drift. In addition there is a lack of appropriate benchmark datasets available. Those available are small or retricted datasets which are quite 'old' in spam terms.

Concept drift is a challenge in spam filtering that has received no research attention to date. It is a considerable challenge which can be handled well by lazy learners, machine learning techniques that do not require the pre-construction of a model which is then used to perform the classification. Lazy learners, such as Case-Based Reasoning (CBR), select the appropriate subset of the training data at the time the request for classification is made and use this subset to perform the classification. They are appropriate techniques for dynamic classification tasks such as spam filtering. Our objective is to apply CBR to the problem of spam filtering. The next chapter discusses machine learning from the viewpoint of CBR with emphasis on the research into associated areas of case-base editing and concept drift.

Chapter 3

CASE-BASED REASONING

Machine Learning addresses the question of contructing computer programs that can improve their performance at a specific task using example data or past experience (Alpaydin 2004, Mitchell 1997). There are many different learning algorithms, techniques and methods that can be applied to a machine learning system but there are two basic requirements that the selected algorithm must balance particularly in the domain of spam filtering; the first is its level of success at generalising from the training data and the second is to perform efficiently.

Incremental or online learning, which can be defined as learning a concept incrementally by processing labelled training examples one at a time (Widmer and Kubat 1993), poses additional challenges. Most significant of these is that the concept being modelled may not be static but can change due to external circumstances or hidden contexts, e.g. weather predictions are influenced by seasonal weather variations, customer buying preferences can be influenced by fashion trends or seasonal inclinations and information filtering is dependent on the document content and user interest. This is known as concept drift and an incremental learner exposed to such concept drift needs to be able to handle it.

In this thesis we are applying machine learning to the problem of spam filtering, specifically applying the machine learning technique of Case-Base Reasoning (CBR) to this problem. This task will require an incremental learning system to process emails one at a time as they arrive at a user's mailbox. Moreover, as discussed previously, concept drift is likely to be a big issue.

The purpose of this chapter is to describe and discuss CBR with particular emphasis on the challenges facing the application of CBR to spam filtering. Two main challenges face the application of CBR to spam filtering; firstly how to manage the training data, the vast collection of emails that exist for most users of email and secondly, how to handle

concept drift.

This chapter starts with a discussion of CBR, describing the process involved and the advantages that it has over other machine learning techniques for certain problem requirements. This section also discusses existing research into producing estimates of classification confidence for predictions in CBR. We then discuss the existing research into case-base management and describe different techniques that could be used to remove ineffective emails from a case-base of training emails. We describe other machine learning techniques that have been used for the spam filtering problem and discuss the different approaches that have been applied to the problem of concept drift in incremental learning problems such as spam filtering.

3.1 Case-Based Reasoning

Case-Base Reasoning (CBR) is a problem solving technique that solves new problems by re-using or adapting solutions that were used to solve similar problems in the past (Riesbeck and Shank 1989). The previous problems or past experience are encoded as cases, each containing features characteristic of the problem and its solution. A collection of these cases, known as the case-base, is the knowledge base of experience used to solve new problems.

One of the strengths of CBR as an alternative problem solving approach is that it can work well in domains that are not well understood. CBR facilitates decision making based on what worked in the past without modelling past decisions in detail.

CBR can be represented as a cyclical process that is divided into the four following sub processes (Aamodt and Plaza 1994) (represented graphically in Figure 3.1):

(i) **retrieve** the most similar case or cases from the case base;

(ii) **reuse** the case to solve the problem;

(iii) **revise** the proposed solution, if necessary;

(iv) **retain** the solution for future problem solving.

A new problem, represented as a case, is compared to the existing cases in the case base and the most similar case or cases are retrieved based on a comparison of case representations. These cases are combined and reused (i.e. adapted) to suggest a solution for the new problem. The solution proposed may need to be revised (i.e. evaluated and

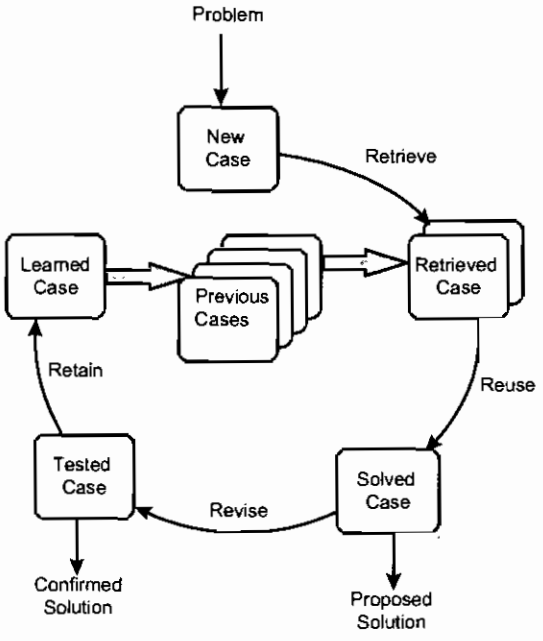


Figure 3.1: The Case-Based Reasoning Process (Aamodt and Plaza, 1994)

corrected) somewhat if it is not a valid solution. This verified solution can be retained by adding it back as a new case to the case base or as amendments to existing cases in the case base for use in future problem solving.

Figure 3.2 (adapted from Cunningham et al. (1994)) illustrates the CBR process in a different way. Consider the target case which represents the problem that is to be solved. The case representation consists of the problem specification and the problem solution. The solution to this target case can be inferred from some first principles reasoning process using the problem specification. The idea in CBR is to avoid modelling this reasoning process and instead to retrieve a similar case from the case-base and adapt its solution to fit the target problem.

3.1.1 Case Representation

"A case is a contextualised piece of knowledge representing an experience" (Watson and Marir 1994). It represents specific knowledge at an operational level. Typically a case includes the problem specification, the solution and sometimes the outcome. While this is the most common representation used, more elaborate case representations can be employed. In Derivational Analogy (DA) (Veloso et al. 1993) cases can encode additional decision making knowledge in the form of a reasoning trace with a view to capturing information about the decision making outcomes and alternatives. DA has been used in

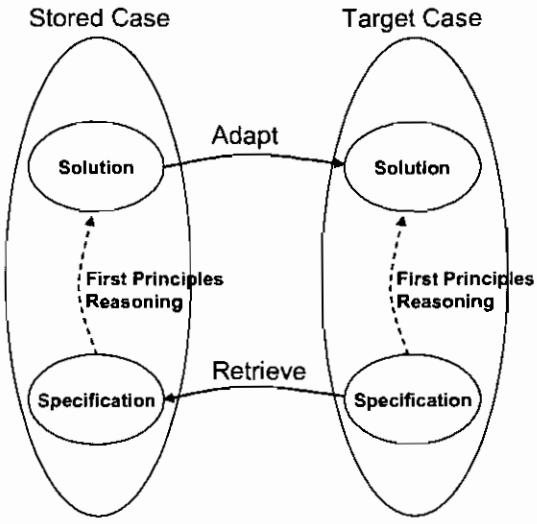


Figure 3.2: The Case-Based Reasoning Process (Cunningham *et al.* 1994)

complex planning tasks where past cases (plans) are re-used by replaying the reasoning traces.

Depending on the information included in a case, different types of results can be achieved from the system. Cases that describe a problem and its solution can be used to derive solutions to new further problems. Those cases that store a problem and its outcome can be used to test 'what-if' scenarios on other new problems whereas the cases that store all three, the problem, its solution and its outcome can be used to evaluate the outcome and prevent potential problems.

In general a case specification is described as a set of features. The features are those aspects of the domain and the problem that are considered to be most significant in determining the solution and/or outcome. The features are identified in a knowledge acquisition exercise that involves interviewing experts in the domain of reference, issuing questionnaires and using other common data gathering techniques. Consider, for example, a credit screening problem. A customer approaches a bank and requests a loan. How does the bank manager decide whether to issue the loan or not? Is this customer a good credit risk or not? This type of problem typically has been implemented using knowledge or rule based systems (more commonly known as expert systems) but it lends itself well to case-based reasoning also. A case represents an experience and in this situation it should represent the features of the application that were used to determine whether to issue the loan or not. These might include the amount of money required, the repayment period, the gender of the applicant, their marital status, their age, their employment status and

employment details such as occupation, salary, etc., the purpose for which the loan is intended (personal or business purchase), and so on.

Table 3.1: Sample case for credit screening problem

Feature	Value
Amount required	2,500
Type of Purchase	Personal
Repayment Period (months)	6
Gender	Male
Age	30
Married	No
Employed	Yes
Weekly wage	€260
Years in Employment	1.5
Recommendation	Accept
Outcome	Good

3.1.2 Case Retrieval

Case retrieval involves finding in the case base the cases closest to the current problem. These cases have the potential to make relevant predictions about the new case. Kolodner (1992) believes that finding the appropriate cases is the core of case-based reasoning.

There are two main retrieval methods in CBR, first is to use a decision tree algorithm and the second is the k-Nearest Neighbour (k-NN) algorithm. The decision tree algorithm (Wess et al. 1994) analyses the features to identify which features do the best job of discriminating between the cases. The most discriminating features are organised into a tree structure with the most discriminating feature placed at the top of the tree. The cases are then organised in memory using this decision tree structure and the retrieval algorithm searches the tree matching the nodes or decisions against the new case to come up with the most appropriate case or cases.

As the cases are arranged in a hierarchical structure the retrieval time for the induction method increases by the logarithm of the number of cases rather than linearly which is more efficient for a large number of cases. This method however requires a significant number of cases to be able to distinguish between the features and identify the appropriate hierarchical structure. This analysis is a time consuming process and must be performed whenever new cases are added to the casebase. Decision trees also do not handle missing feature values very well.

The k-NN retrieval algorithm compares each case in the case base with the target case and calculates a similarity metric for each selected case. This similarity metric is based on how 'close' the features of the selected case are to those of the new case. Each feature is compared and a score is assigned depending on the difference between the two features; the closer the features the higher the score. Weights can be assigned to the features depending on their relevance to the solution. The k cases with the highest similarity values will be retrieved. The solution of the target case can then be derived from these k neighbour cases. If the solution is a class, for instance, majority voting can indicate the class of the target case.

The disadvantage of this method is that the search time increases linearly with the number of cases in the case base. An improvement to this was introduced by Lenz et al. (1998a) who introduced a similarity-based algorithm called Case Retrieval Nets (CRN). A CRN is a memory structure which allows an efficient yet flexible retrieval of cases. It borrows ideas from neural networks and associative memory models. It is made up of the following components:

- *Case nodes* represent stored cases.
- *Information Entity Nodes (IEs)* represent feature-value pairs within cases
- *Relevance Arcs* link case nodes with the IEs that represent them. They have weights that capture the importance of the IE.
- *Similarity Arcs* connect IEs that refer to the same features, and have weights relative to the similarity between connected IEs.

The idea behind the CRN architecture is that a target case is 'activated' by connecting it to the net via a set of relevance arcs and this activation is then spread across the net. Each of the other case nodes accumulates an activation score appropriate to its similarity to the target case. The case nodes with the highest activation are the most similar cases to the target case. CRNs take advantage of redundancy in feature values and they are tolerant of missing or unknown feature values.

3.1.3 Reuse

In situations where the retrieved case is identical to the target case the retrieved case can be used as is and the solution applied. Because new situations rarely match old ones typically there may not be an exact match for a new case. In this situation old solutions

are altered to reflect the problem specifications of the target case; this process is called adaptation. Several adaptation techniques are used in CBR which can be described as a continuum of adaptation models (Wilke and Bergmann 1998, Wilke et al. 1998), as illustrated in Figure 3.3.

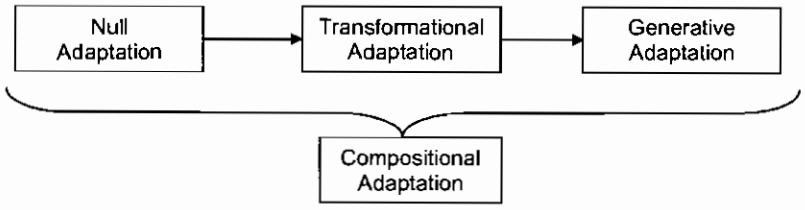


Figure 3.3: The Continuum of Adaptation Techniques, adapted from Wilke and Bergmann (1998)

The simplest, known as *null adaptation* is the situation when no adaptation is necessary and the solution can be applied directly, this is common in classification problems. Transformational adaptation employs a set of rules that modify the retrieved solution based on differences between the features of the retrieved and target cases. Generative models (also known as Derivational Analogy) are significantly more complex and require a generative from-scratch problem solver that is tightly integrated into the CBR system. The problem solver is used to generate, from scratch, those parts of the solution that are inadequate due to the differences between the features of the retrieved and target cases. Compositional techniques involve creating a composite solution to the target case by combining newly adapted solution components from multiple retrieved cases.

Complex adaptation poses a number of problems. Watson (1997) argues that complex adaptation is knowledge intensive and with CBR used for problem areas that are not well-understood, the knowledge necessary for defining tranformational rules or generative from-scratch processes may not be readily available or understood. It has also been shown in domains requiring more complex adaptation techniques such as planning, that reuse can be more complex that generation from first principles (Nebel and Koehler 1995).

3.1.4 Revision and Retension

The final two processes in the CBR cycle can be considered together. These processes together constitute the learning process of case-based reasoning. An opportunity for learning arises when the solution output from the reuse process is not correct. The solution must be revised (modified) to a correct solution and if appropriate retained or added to the case

base to allow it to influence further problems. Opportunities for learning also arise when CBR systems have been used for speed-up learning. In this situation the benefit of CBR is that it allows the discovery of short cuts in the problem solving process by the reuse of previous solutions (Veloso et al. 1993) .

The process of revision involves two steps, the evaluation of the solution and the diagnosis and repair of the fault, if necessary. The evaluation step involves judging how good is the solution provided by the case basc. This evaluation can be done in a number of ways. It can be based on feedback from the real world, either by asking the expert or by testing the solution in the real environment. Evaluation can also be based on results of simulations of applying the solution. The outcome of the evaluation could suggest that further adaptation or repair is necessary. The revised case is then available for addition to the case base.

The retention process is the augmenting of the case base with the revised case thereby allowing the system to learn. The learning occurs as a by-product of the problem solving process and represents learning by experience. Successful new solutions are added to the case memory to facilitate similar problems in the future and failures can be added to avoid repeating the same mistakes.

Increasing the number of example situations in the case base means that the system covers more of the domain and will work better. However, increasing the size of the case-base indiscriminately leads to a problem known as the *utility problem.* After a certain point, adding more cases to a case-base will result in a reduction in the efficiency of the case-base (Smyth and Cunningham 1996). This comes from the fact that retrieval time will increase but the adaptation savings tend to reduce as more cases are added. This has led to research in the area of case-base editing which is discussed in detail below in Section 3.2. One central focus in case-base editing involves reducing the size of the case-base while maintaining the performance.

3.1.5 Advantages of CBR

Case-based reasoning offers advantages over other machine learning techniques. CBR is a lazy learning technique (Aha 1997). *Lazy* learners delay processing until problem solving time whereas *eager* learners construct a model from the training examples in advance of any problem solving requests. Eager learners then use the model itself to process the requests. The advantage that lazy learners have over eager learners is that new training examples can be added to the system very easily. It is particularly suited to situations

where training examples are not necessarily available at the beginning but are collected online. The limitations of lazy learning are that all the examples used need to be stored and accessible to the system for each request. Also, it is more computationally expensive at run time to perform the processing at the time of the request. However, the rapid evolution of computer hardware continues to ease these limitiations.

Lazy learners also belong to the class of local learning algorithms (Bottou and Vapnik 1992). The lazy learner selects appropriate and relevant training examples and effectively builds a local model for each request that it processes. This offers further advantages over eager techniques in domains where there is a lack of homogeneity among the training examples. If the training examples cover a domain where there is variety in the types of examples, a global learner will attempt to build a model that encapsulates the different types. A local learner, on the other hand, can just use the training examples of a specific type which are most relevant to the request to provide a solution for that specific request. Local learners also do not suffer from data interference (Atkeson et al. 1997). New examples being added to the training set in one area do not affect performance in others.

3.1.6 Estimates of Confidence in CBR

Cheetham and Price emphasise the importance of being able to attach confidence values to predictions in CBR (Cheetham 2000, Cheetham and Price 2004). This has been a research issue since the earliest days of expert systems research: it is part of the body of research on meta-level knowledge (Lenat et al. 1983, Davis and Buchanan 1985), the view being that it is important for a system to 'know what it knows'. TEIRESIAS is a system in this spirit, it was designed to simply admit its ignorance instead of venturing risky advice (Davis 1982).

More recently, the system SIROCCO from McLaren and Ashley (2001) uses meta-rules to determine the system's confidence. Their system operates in an engineering ethics domain, in which incorrect suggestions could be considered sensitive and damaging. In this system, if any one of the meta-rules are fired then the system considers itself inadequate for the task. Their evaluation of SIROCCO shows that allowing the system to produce 'don't know' results reduces the number of incorrectly classified cases, with a small trade off whereby the number of correctly classified cases is reduced.

Reilly et al. (2005) present a model of confidence designed with conversational recommender systems in mind, specifically those employing critiquing as their primary source

of feedback. Their strategy is based on modelling confidence at a feature level with the individual feature confidence values contributing to a measure of case confidence for each case returned as a recommendation. They found that their confidence-based recommendation strategy improves recommendation efficiency, producing shorter and therefore more successful sessions.

So while it is clear that it is useful to be able to produce estimates of confidence, it is also clear that generating reliable estimates is not straightforward. Cheetham and Price (2004) describe 12 measures of confidence that can be applicable for a k-NN classifier. Some of these indicators increase with confidence and some decrease. Since no single indicator is capable of producing a robust measure of confidence they explore the use of a decision tree, that is allowed to use all the measures, as a mechanism for aggregating all the available metrics. The authors show that, even using a decision tree to learn a good confidence measure from historic data, it is difficult to avoid the situation where predictions labelled as confident prove to be incorrect. They also emphasise that the confidence estimation mechanism will need to be updated over time as the nature of the problems being solved can change.

Predicting estimates of confidence is relevant for spam filtering because a system capable of handling concept drift will need user intervention to indicate when mistakes have been made. If an estimate of the prediction confidence can be made with the prediciton it may remove some of the onus on the user of checking all the system's predictions; the user being able to ignore those predictions that are made with confidence.

3.1.7 Textual CBR

An area of research in CBR that is relevant to spam filtering is the general area of Textual Case-Based Reasoning (TCBR) (Lenz et al. 1998b). TCBR is an area of CBR that deals with cases which represent textual documents. There are a number of domains where TCBR has been used. These include help desks (Lenz 1998, Lenz et al. 1998b), customer support (Gupta and Aha 2004), intelligent tutoring (Ashley and Aleven 1991) and law (Bruninghaus and Ashley 2001, 2003).

Textual CBR offers some advantages over Information Retrieval (IR) (Baeza-Yates and Ribeiro-Neto 1999), the more traditional approach for identifying relevant documents. IR normally identifies the set of index terms (normally words) of the documents in the collection using statistical means, i.e. frequencies or probabilities of occurrence of a particular word in the document. These frequencies are also used to determine the importance of the

feature in the document and consequently used in the calculation of similarity between documents. These ideas have been used in TCBR too, but using CBR, domain specific knowledge can relatively easily be included in the case representation, e.g. keyword combinations or document structure. An illustration of this is the use of IR in the legal domain, where term frequencies do not take account of domain specific knowledge and therefore can only retrieve approximately 25% of the relevant documents (Blair and Maron 1990). TCBR is also normally applied to a specific domain which can help avoid the problem of ambiguity, where the same word can be used in more than one document but with a different meaning.

In TCBR, cases have to be extracted from the textual documents and the representation of these textual documents is key as it is used to compute the similarity between cases. Case representations in TCBR can be more sophisticated that those used in IR. They can include shallow natural language processing techniques such as Part-of-Speech tagging and structured information in the form of attribute-value pairs (Lenz et al. 1998b), or domain specific thesaurus (Bruninghaus and Ashley 1999), and even latent semantic analysis which extends the case representation based on the semantic similarity of words (Foltz et al. 1999). A characteristic of TCBR is therefore, the potential to be more 'knowledge based' than IR. However, the case-based reasoning approach used in this thesis uses the vector-space model that is characteristic of IR systems.

3.2 Case-base editing

An area of CBR that has prompted much recent research is case-base editing, which involves reducing the number of previous problems used in the knowledge base while maintaining or even improving performance. It is of particular interest for the spam filtering domain as each email is effectively a case and an individual can receive large volumes of email, so an issue that needs to be addressed is which cases should be incorporated into the knowledge base used to classify new emails. There is significant research in this area which is described in this section.

3.2.1 Early Techniques

Case-base editing techniques have been categorised by Brighton and Mellish (2002) as competence preservation or competence enhancement techniques. Competence preservation corresponds to redundancy reduction, removing superfluous cases that do not contribute

to classification competence. Competence enhancement is effectively noise reduction, removing noisy or corrupt cases from the training set. Figure 3.4 illustrates both of these, where cases of one class are represented by stars and cases of the other class are represented by circles. Competence preservation techniques aim to remove internal cases in a cluster of cases of the same class and can predispose towards preserving noisy cases as exceptions or border cases. Noise reduction on the other hand aims to remove noisy or corrupt cases but can remove exceptional or border cases which may not be distinguishable from true noise, so a balance of both can be useful.

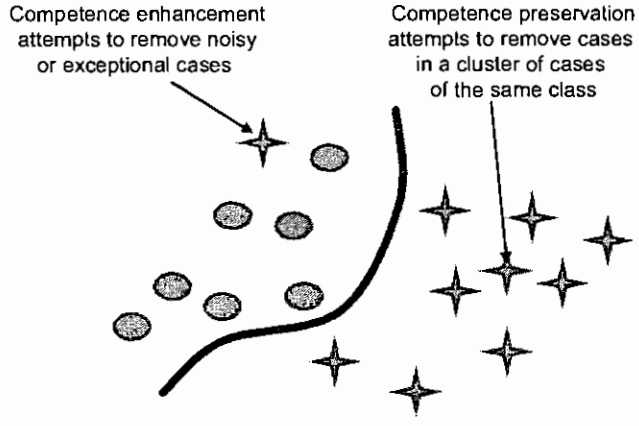


Figure 3.4: Case-base editing techniques demonstrating competence preservation and competence enhancement

Editing strategies normally operate in one of two ways; *incremental* which involves adding selected cases from the training set to an initially empty edited set, and *decremental* which involves contracting the training set by removing selected cases.

An early competence preservation technique is Hart's Condensed Nearest Neighbour (CNN) (Hart 1968). CNN is an incremental technique which adds to an initially empty edited set any case from the training set that cannot be classified correctly by the edited set. This technique is very sensitive to noise and to the order of presentation of the training set cases, in fact CNN by definition will tend to preserve noisy cases. Ritter et al. (1975) reported improvements on the CNN with their Selective Nearest Neighbour (SNN) which imposes the rule that every case in the training set must be closer to a case of the same class in the edited set than to any other training case of a different class. Gates (1972) introduced a decremental technique which starts with the edited set equal to the training set and removes a case from the edited set where its removal does not cause any other training case to be misclassified. This technique will allow for the removal of noisy cases

but is sensitive to the order of presentation of cases.

Competence enhancement or noise reduction techniques start with Wilson's Edited Nearest Neighbour (ENN) algorithm (Wilson 1972), a decremental strategy, which removes cases from the training set which do not agree with their k nearest neighbours. These cases are considered to be noise and appear as exceptional cases in a group of cases of the same class.

Tomek (1976) extended this with his repeated ENN (RENN) and his *all k-NN* algorithms. Both make multiple passes over the training set, the former repeating the ENN algorithm until no further eliminations can be made from the training set and the latter using incrementing values of k. These techniques focus on noisy or exceptional cases and do not result in the same storage reduction gains as the competence preservation approaches.

Later editing techniques can be classified as hybrid techniques incorporating both competence preservation and competence enhancement stages. Aha et al. (1991) presented a series of instance based learning algorithms to reduce storage requirements and tolerate noisy instances. IB2 is similar to CNN adding only cases that cannot be classified correctly by the reduced training set. IB2's susceptibility to noise is handled by IB3 which records how well cases are classifying and only keeps those that classify correctly to a statistically significant degree. Other researchers have provided variations on the IBn algorithms (Brodley 1993, Cameron-Jones 1992, Zhang 1992).

3.2.2 Competence-Based Case-Base Editing

More recent approaches to case-base editing build a competence model of the training data and use the competence properties of the cases to determine which cases to include in the edited set. Measuring and using case competence to guide case-base maintenance was first introduced by Smyth and Keane (1995) and developed by Zu and Yang (1997). Smyth and Keane (1995) introduce two important competence properties, the *reachability* and *coverage* sets for a case in a case-base. The *reachability set* of a case c is the set of all cases that can successfully classify c, and the *coverage set* of a case c is the set of all cases that c can successfully classify. These are discussed in Chapter 5. The coverage and reachability sets represent the local competence characteristics of a case and are used as the basis of a number of editing techniques.

McKenna and Smyth (2000) presented a family of competence-guided editing methods for case-bases which combine both incremental and decremental strategies. The family of

algorithms is based on four features;

(i) an *ordering policy* for the presentation of the cases that is based on the competence characteristics of the cases,

(ii) an *addition rule* to determine the cases to be added to the edited set,

(iii) a *deletion rule* to determine the cases to be removed from the training set and

(iv) an *update policy* which indicates whether the competence model is updated after each editing step.

The different combinations of ordering policy, addition rule, deletion rule and update policy produce the family of algorithms.

Brighton and Mellish (2002) also use the coverage and reachability properties of cases in their Iterative Case Filtering (ICF) algorithm. ICF is a decremental strategy contracting the training set by removing those cases c, where the number of other cases that can correctly classify c is higher that the number of cases that c can correctly classify. This strategy focuses on removing cases far from class borders. After each pass over the training set, the competence model is updated and the process repeated until no more cases can be removed. ICF includes a pre-processing noise reduction stage, effectively RENN, to remove noisy cases. McKenna and Smyth compared their family of algorithms to ICF and concluded that the overall best algorithm of the family delivered improved accuracy (albeit marginal, 0.22%) with less than 50% of the cases needed by the ICF edited set (McKenna and Smyth 2000).

Wilson and Martinez (1997) present a series of Reduction Technique (RT) algorithms, RT1, RT2 and RT3 which, although published before the definitions of coverage and reachability, could also be considered to use a competence model. They define the set of associates of a case c which is comparable to the coverage set of McKenna and Smyth except that the associates set will include cases of a different class from case c whereas the coverage set will only include cases of the same class as c. The RTn algorithms use a decremental strategy. RT1, the basic algorithm, removes a case c if at least as many of its associates would still be classified correctly without c. This algorithm focuses on removing noisy cases and cases at the centre of clusters of cases of the same class as their associates which will most probably still be classified correctly without them. RT2 fixes the order of presentation of cases as those furthest from their nearest unlike neighbour (i.e. nearest case of a different class) to remove cases furthest from the class borders first. RT2

also uses the original set of associates when making the deletion decision, which effectively means that the associate's competence model is not rebuilt after each editing step which is done in RT1. RT3 adds a noise reduction pre-processing pass based on Wilson's noise reduction algorithm.

Wilson and Martinez (1997) concluded from their evaluation of the RTn algorithms against IB3 that RT3 had a higher average generalization accuracy and lower storage requirements overall but that certain datasets seem well suited to the techniques while others were unsuited. Brighton and Mellish (2002) evaluated their ICF against RT3 and found that neither algorithm consistently out performed the other and both represented the "cutting edge in instance set reduction techniques" .

3.3 Concept Drift

In a number of real life situations the concept being modelled is not static but is subject to concept drift. The task of a machine learning system is to model the concept reflected in the training data and to use this model to induce solutions to new problems as they are presented. An incremental learner exposed to such concept drift needs to be able to handle it.

An analysis of the machine learning literature on concept drift suggests that there are three general approaches; instance selection, instance weighting and ensemble learning (Tsymbal 2004) which are discussed below.

3.3.1 Instance Selection

The goal of instance selection is to select instances that are representative of the current concept. The most common concept drift technique is based on instance selection and involves maintaining a time *window* on the training data. The window slides over the recent training instances, using the most recent for prediction of later instances. Examples of window-based algorithms include the FLORA family of algorithms (Widmer and Kubat 1996), FRANN (Kubat and Widmer 1995) and Time-Windowed Forgetting (TWF) (Salganicoff 1997). Some algorithms use a window of fixed size while others adjust the window size based on the speed and amount of drift, e.g. "Adaptive Size" (Klinkenberg 2004) and FLORA2 (Widmer and Kubat 1996). Another instance selection technique called *batch selection* is proposed by Klinkenberg (2004) which selects batches of instances based on their performance on the newest batch of instances. In general, the window-based algo-

rithms have been shown to be successful at speeding up the process of relearning concepts and at quickly adapting to new concepts.

3.3.2 Instance Weighting

Instance weighting uses the ability of some learning algorithms such as Support Vector Machines (SVMs) to process weighted instances (Klinkenberg 2004). Instances can be weighted according to their age and/or their performance on the training examples. Klinkenberg (2004) shows that the window and batch selection techniques handle concept drift better than analogous instance weighting techniques which is possibly due to overfitting the data.

3.3.3 Ensemble Learning

An ensemble learner combines the results of a number of classifiers, where each base (component) classifier is constructed on a subset of the available training instances. The research issues involved in using an ensemble for handling concept drift involve firstly determining how to partition the instances into subsets with which to train the base classifiers. Then a mechanism for aggregating the results of the base classifiers must be identified. Finally, a mechanism for updating the ensemble to handle new instances and "forget" older instances must be established.

Building on the analysis presented in Kuncheva (2004) we propose that the techniques for using ensembles to handle concept drift fall into two groups:

- dynamic combiners where the base classifiers are trained in advance and the concept drift is tracked by changing the aggregation rule,
- incremental approaches that use fresh data to update the ensemble and incorporate a "forgetting" mechanism to remove old or redundant data from the ensemble.

These approaches will be discussed below. It is worth noting that the two approaches are not mutually exclusive and combinations of both are possible.

Dynamic Combiners

The main techniques used for the dynamic combiners are variants on the Weighted Majority algorithm (Littlestone and Warmuth 1994) where the weights on the base classifiers are altered according to how the base classifier performs as compared with the overall

ensemble result. The issue with dynamic combiners is that the base classifiers are not re-trained with new instances so this approach is not appropriate for incremental learners exposed to new contexts as it is necessary to create new ensemble members to handle the new contexts. Dynamic Weighted Majority (Kolter and Maloof 2003) attempts to resolve this problem using the Weighted Majority algorithm and combining it with an update policy to create and delete base classifiers in response to changes in performance.

Incremental Ensembles

The decision on how to partition the data into subsets with which to train the base classifiers is sometimes termed 'data selection'. This decision will also determine how fresh instances are added into the ensemble. Kuncheva (2004) categorises three data selection approaches. The first reuses data points as is done in Bagging (random sampling with replacement) (Breiman 1996). The second approach to data selection is a filtering approach as in Boosting (Freund and Schapire 1999) or that used by Breiman (Breiman 1999). The final data selection approach and the most common approach is one which uses blocks or chunks of data. These blocks normally group the data sequentially and could be of fixed size (Street and Kim 2001, Wang et al. 2003) or of variable size (Kolter and Maloof 2003, Stanley 2003).

Any incremental ensemble approach requires a *forgetting* mechanism to identify which base classifiers should be dropped from the ensemble as the new members are added. The simplest forgetting strategy is to drop the oldest classifier once a new member has been added. More complex strategies are based on the actual performance of the base classifiers. Wang et al. (2003) keeps the top K base classifiers with the highest accuracy on the current training data chunk while Street and Kim (2001) favour the base classifiers that correctly classify instances (of the current block) on which the ensemble is 'nearly undecided'. The worst performing classifer is replaced by the new member classifier. Stanley (2003) and Kolter and Maloof (2003) record the performance of each member against all seen instances and periodically remove those classifiers whose performance falls below a particular threshold.

3.4 ML techniques for spam filtering

Existing machine learning research into spam filtering can be categorised by the classification technique used. The main techniques applied to the problem are Naïve Bayes and

Support Vector Machines. This section discusses these techniques and describe how they operate.

3.4.1 Naïve Bayes

Naïve Bayes is a probabilistic classifier that can handle bigh dimensional data that can be a problem for other machine learning techniques. It is 'naïve' in the sense that it assumes that the features are independent. Consider a group of documents that are labelled as one of a set of classifications $c_i \in C$. Each document is described by a set of attributes $\{a_1, a_2, \ldots a_n\}$ where a_i indicates the presence of that attribute in the document. The classification returned from a Naïve Bayes classifier for a new document is given in Equation 3.1.

$$c_{NB} = \underset{c_i \in C}{\operatorname{argmax}} P(c_i) \prod_j P(a_j|c_i) \tag{3.1}$$

Due to the significance of false positives (legitimate emails identified incorrectly as spam) in spam filtering, the Naïve Bayes classifier is not generally used in this simple *argmax* form. In practice the classification threshold is set to bias the classifier away from false positives (see Section 5.3.1).

The conditional probabilities can be estimated by $P(a_i|c_j) = n_{ij}/n_j$ where n_{ij} is the number of times that attributes a_i occurs in those documents with classification c_j and n_j is the number of documents with classification c_j. This provides a good estimate of the probability in many situations but in situations where n_{ij} is very small or even equal to zero this probability will dominate, resulting in an overall zero probability. A solution to this is to incorporate a small-sample correction into all probabilities called the Laplace correction (Niblett 1987). The corrected probability estimate is given by Equation 3.2, where n_{ki} is the number of values for attribute a_i. Kohavi et al. (1997) suggest a value of $f = 1/m$ where m is equal to the number of training documents.

$$P(a_i|c_j) = \frac{n_{ij} + f}{n_j + f n_{ki}} \tag{3.2}$$

Existing research into the application of Naïve Bayes to spam filtering, which was discussed in Section 2.4.2, reports good performance on static datasets. There is good corroborative evidence that Naïve Bayes performs better than rule-based techniques and at least as well as memory based techniques. However isolated studies have argued that it can be outperformed by decision trees and neural networks although there is an ac-

knowledgment of the impractical training times associated with learners such as neural networks.

3.4.2 Support Vector Machines

A Support Vector Machine (SVM) (Christianini and Shawe-Taylor 2000, Vapnik 1999) is a linear maximal margin binary classifier. It can be interpreted as finding a hyperplane in a linearly separable feature space that separates the two classes with maximum margin (see Figure 3.5(a)) The instances closest to the hyperplane are known as the "support vectors" as they support the hyperplane on both sides of the margin.

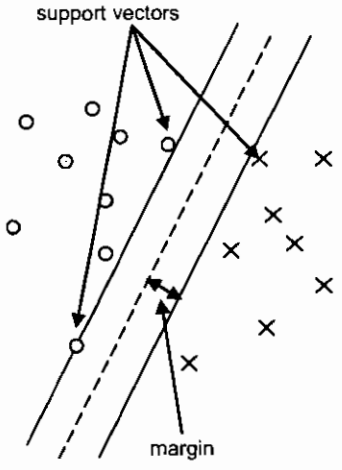


Figure 3.5: An SVM classifier

An SVM is the solution to a quadratic dual optimisation problem that uses the inner product of the instances in the original feature space. Often the original feature space is not linearly separable and is mapped to a higher dimension feature space. Combining the transformation and the inner product into a single operation (known as the Kernel function) allows the SVM to use instances in the original feature space without the explicit calculation of the transformation. In the case of text, the dimensionality of the input feature space is large enough for the simple dot product to be used as the Kernel function (Zamolotskikh et al. 2006).

A hard-margin SVM attempts to solve the optimisation problem finding a hyperplane that completely separates the two classifications. Since noise or inseparable data can make it impossible to find a hard margin, the soft margin SVM 'softens' the constraints in the optimisation problem by adding slack variables. This allows for noise in the data (e.g. some data points to fall within the margin or even on the wrong side of the margin).

A 1-norm or a 2-norm soft margin SVM are different ways to approach the soft margin SVM. In the 1-norm case, the optimisation problem's goal function uses a sum of the slack variables while for the 2-norm case, the goal function uses a sum of squares of slack variables.

Existing research into spam filtering using SVMs, discussed in Section 2.4.2, has shown that they also perform well on static datasets. A problem with SMVs in a dynamic situation is that they are difficult to update with new examples of training data.

3.5 Conclusions

In this chapter we reviewed a machine learning approach of case-base reasoning discussing the 4-R model (retrieve, reuse, revise and retain) which is commonly used. As a lazy, local learner CBR offers distinct advantages over other eager or global approaches to the problem of spam filtering. As spam filtering suffers from concept drift, it is very important to be able to keep the system up to date with examples of new types of spam as it becomes available. A CBR system facilitates this as all that is needed is to simply add the new examples into the case-base. Also, spam is a diverse concept. Pornographic spam has little in common with spam offering cheap mortgages, or spam advertising cheap drugs. Even an individual's legitimate email will include a variety of different types, personal, work-related and hobby or interest-focussed email. CBR as a local learner will use the most appropriate subset of all training emails to classify new examples.

This chapter also discusses existing research in two associated areas; case-base editing techniques and mechanisms for handling concept drift. Both are pertinent to spam filtering. Case-base editing techniques are of interest due to the volume of emails available from which to select appropriate training data. In the following chapter we describe a new case-base editing technique which uses the competence properties of the cases in the case-base to determine which cases to include in the training set. We then show in Chapter 5 how this technique helps in the filtering of spam email.

Mechanisms for handling concept drift, also discussed in this chapter, are relevant as we will also require a technique for handling the concept drift in email to effectively filter spam. In our discussion of handling concept drift we categorised the main approaches as instance selection, instance weighting and ensemble learning. The technique we propose for handling concept drift in spam and legitimate email, introduced in the next chapter, falls into the category of instance selection although it doesn't follow the most common

technique applied, that of windowing. However, in chapter 6 we compare our approach, which is effectively a case-base maintenance approach, with that of windowing and also ensemble learning.

The next chapter presents the design of our case-based system to filter spam discussing the case representation, feature selection and the case-base editing used. It also describes the architecture of the online real-time system that was implemented to evaluate the research ideas presented in this thesis.

Chapter 4

SYSTEM DESIGN

This chapter describes the design of the case-based spam filtering system which we call Email Classification Using Examples (ECUE). The chapter starts with a description of the design of the case-base used by ECUE. It discusses the design decisions related to the design of the case-base itself, specifically the feature extraction, feature selection and feature representation for the cases in the case-base and how case retrieval and case-base editing is performed on the case-base (Delany et al. 2004a, 2005a).

A user of ECUE may receive over a hundred legitimate emails a week and a multiple of that in spam. So there is an ongoing need to discard emails that are not contributing to the competence of the filter. In addition spam email (and to some extent legitimate email) is subject to concept drift so there is a need to allow the filter to use additional examples of email as new training cases to maintain the competence of the filter. This chapter outlines the case-base maintenance approach that ECUE uses (Delany et al. 2004b, 2005b) to handle these issues. This case-base maintenance strategy has two components; a case-editing algorithm to remove noisy and exceptional cases and a case-base update policy to allow the addition of new training examples to the case-base as they are encountered. This chapter includes a detailed description of the Competence Based Editing algorithm devised to edit the case-base and describes the case-base update policy used to add new examples of spam and legitimate email to the case-base.

The chapter concludes with the design of the online system that was implemented to perform the real-time evaluation of the spam filtering application. This includes a description of the technical architecture designed to integrate with the mail server to allow emails to be captured for filtering and a description of the processes that implement the actual filtering functionality, including case-base setup, filtering, learning and reporting.

4.1 Case-Base Design

This section outliues the case-base design, specifically the features of the individual cases. It discusses how features are extracted from email messages and what features are extracted. It describes the feature selection process used to select those features that are most predictive of spam and legitimate emails and shows how cases/features are represented in the case-base. It discusses how cases are retrieved from the case-base to be used in classification and how classification is performed discussing the design decisions behind the k-nearest neighbour classifier that is used. Lastly it outlines the case-editing technique.

4.1.1 Feature Extraction

In order to identify the possible lexical features from the training set of emails, each email was parsed and tokenised. No stop word removal, stemming or lemmatisation was performed on the text before tokenisation. Email attachments were removed before parsing but any HTML text preseut in the email was included in the tokenisation. The datasets used throughout the evaluations were personal datasets, i.e. all emails in each dataset were sent to the same individual. Hence it was felt that certain headers may contain useful information so a selection of header fields, including the Subject, To and From headers were included in the tokenisation. This is supported by a number of researchers (Drucker et al. 1999, Rennie 2000, Zhang et al. 2004) who concluded that the information from the message header is as important as the message body.

Three types of features were identified:

- word features (i.e. sequences of characters separated by white space or separated by start and end HTML tag markers);
- letter or single character features;
- statistical features, including the proportion of uppercase, lowercase, punctuation and whitespace characters.

4.1.2 Feature Representation

In a case-based learner, examples in the training data are represented as cases in a case-base. For the spam filtering domain, each training example is a case e_i represented as a vector of feature values, $e_j = (f_{1j}, f_{2j}, \ldots f_{nj}, s)$. In text classification the lexical features

are normally represented in one of two ways: (a) binary i.e. if feature f_i exists in email e_j, $f_{ij} = 1$, otherwise $f_{ij} = 0$, or (b) numeric where f_{ij} is a number indicative of the frequency of occurrence of the feature in the email. Feature s represents the class of the email, in our situation either *spam* or *nonspam*.

For numeric features a common way to determine the value of f_{ij} for feature f_i in email e_j is to use the normalised frequency of the feature, see Equation 4.1, where $freq_{ij}$ is the number of times that feature f_i occurs in email e_j.

$$f_{ij} = \frac{freq_{ij}}{\max_k freq_{kj}} \tag{4.1}$$

In the evaluations to determine the most appropriate feature representation we used this normalised frequency for word and letter features (separate normalisations for each type). In addition, we used the actual calculated proportion for the statistical features which is between zero and one by definition.

A binary representation for the different types of feature is not so straightforward. For word features we use the simple existence rule that if the word exists in the email the feature value $f_{ij} = 1$ otherwise $f_{ij} = 0$. However for letter features, almost all letters or characters will occur within an email so using the existence rule is not useful. For letter features we use the Information Gain (Quinlan 1997) value of the feature as calculated during the feature selection process (see Section 4.1.3) to determine whether to set $f_{ij} = 1$. If the normalised frequency of the letter feature is greater than or equal to the normalised frequency which returns the highest information gain for that letter then the feature value is set to one in the case representation, otherwise it is zero. Given that statistical features are also values between zero and one, this rule was also applied to features of this type to determine their binary representation.

Figure 4.1 show the improvements in accuracy achieved by including a binary representation for letters over the more standard frequency measure.

It is more normal in text classification for lexical features to carry frequency information but the results of our evaluations (see Section 5.2) concluded that for this domain the performance implications of using numeric features over binary features outweighed the minor improvement in accuracy achieved by using numeric features especially considering that there were no significant improvements in the rate of FPs.

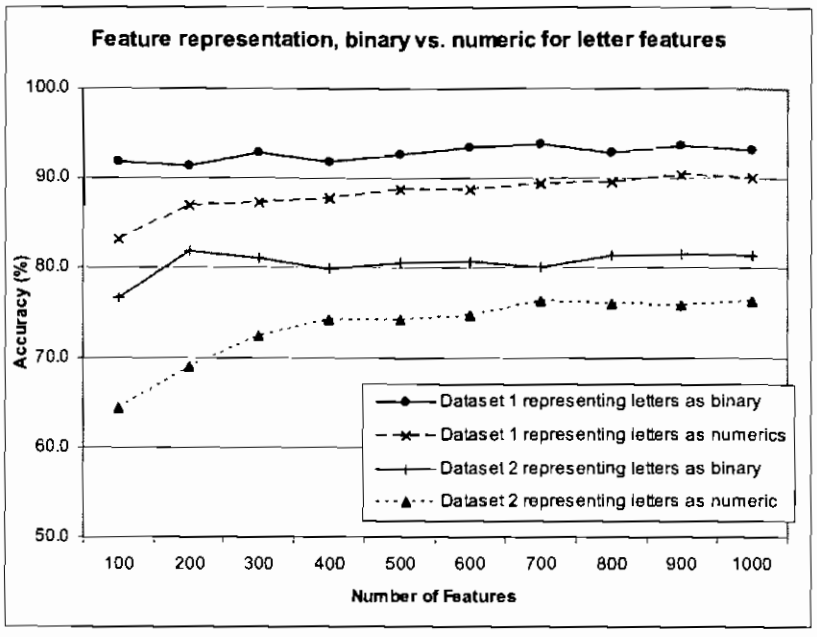


Figure 4.1: Binary letter feature representation compared with numeric letter feature representation. Experiments involved 10-fold cross validation on two datasets of 1000 emails, 500 spam and 500 legitimate.

4.1.3 Feature Selection

Tokenising one thousand emails results in a very large number of features; tens of thousands of features. Feature selection is necessary to reduce the dimensionality of the feature space. Yang and Pedersen (1997)'s evaluation of dimensionality reduction in text categorisation found that Information Gain (IG) (Quinlan 1997) was one of the top two most effective techniques for aggressive feature removal without losing classification accuracy.

The Information Gain of a feature is a measure of the amount of information that a feature brings to the training set. It is defined as the expected reduction in entrophy caused by partitioning the traiuing set T using the feature A as shown in Equation 4.2 where T_v is that subset of the training set T that has feature A equal to value v.

$$IG(T, A) = Entropy(T) - \sum_{v \in values(A)} \frac{|T_v|}{|T|} Entropy(T_v) \tag{4.2}$$

Entropy is a measure of how much randomness or impurity there is in the data set. It is defined in Equation 4.3 where c equals the number of classes in the training set.

$$Entropy(T) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{4.3}$$

In our case, as we are dealing with two classifications *spam* and *nonspam*, the $Entropy(T)$

can be simplified to $Entropy(T) = -p_+ \log_2 p_+ - p_- \log_2 p_-$ where p_+ represents the *spam* class and p_- the *nonspam* class.

The feature selection technique used by Cunningham et al. (2003) when using CBR for spam filtering was Odds Ratio (OR)(Mladenic 1998). OR is a feature selection technique for a binary classifier which uses the ratio of the odds of a feature occuring in one classificatioan to the odds of the feature occuring in the other classification. We compared using IG with OR where, for convenience, we calculated the OR for feature f_i occuring in classification c_j (i.e. *spam* and *nonspam*) as given in Equation 4.4 below. Where a specific feature did not occur in a classification, we assigned it a small fixed value frequency so that the OR value can still be calculated.

$$OR(f_i, c_j) = \frac{P(f_i|c_j)}{P(f_i|\bar{c_j})} \tag{4.4}$$

In these experiments we selected the n features with the highest IG value and $\frac{n}{2}$ features each from $OR(f_i, spam)$ and $OR(f_i, nonspam)$ sets. The results, displayed in Figure 4.2, showed that IG performed significantly better than OR. It is worth noting that there were a significant number of zero activations i.e. target cases with zero similarity to any case in the case-base, for the OR selection technique. 8.8% compared with 0.2% for the IG technique. This is due to the tendency for OR to select rare terms.

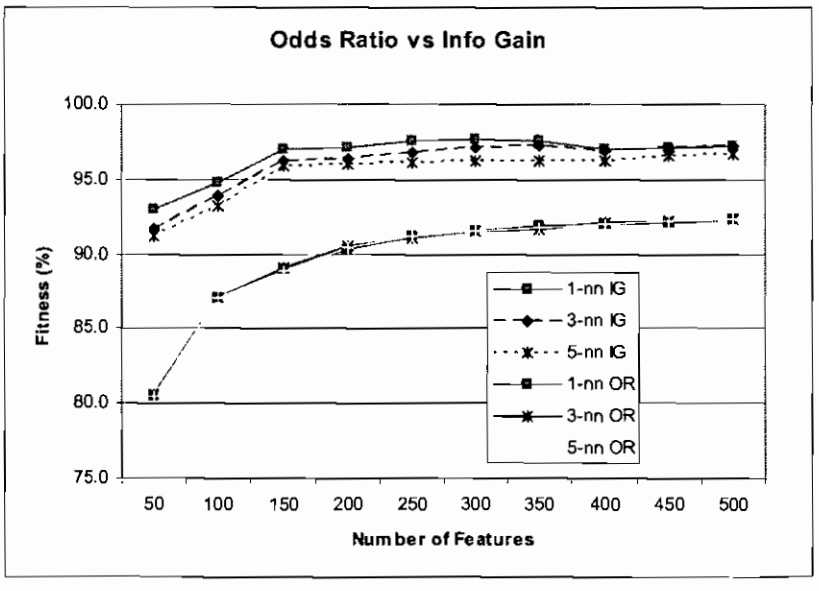


Figure 4.2: Comparing Information Gain with Odds Ratio. Results of the average of three 10-fold cross validation experiments on a dataset of 1000 emails, 500 spam and 500 legitimate where word features only were used.

We calculated the IG of each feature and the top 700 features were selected. Our cross

validation experiments, varying between 100 and 1000 features across 3 datasets, indicated best performance at 700 features, see Figure 4.3.

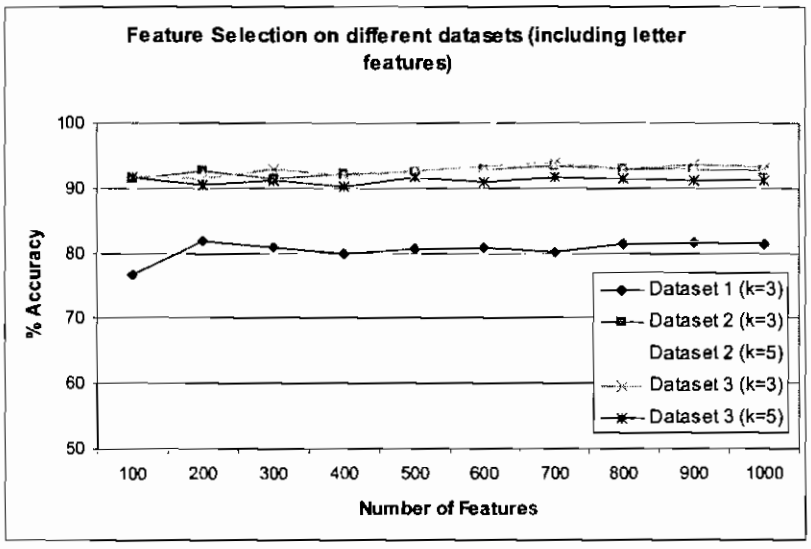


Figure 4.3: Results of cross validation experiments to identify the most appropriate number of features to use. Experiments involved 10-fold cross validation on three datasets of 1000 emails, 500 spam and 500 legitimate.

4.1.4 k-Nearest Neighbour Classifier

A k-Nearest Neighbour (k-NN) classifier assigns a classification to a target case by identifying and analysing the training cases that are most similar to it. The similarity measure Sim between target case e_t and case-base case e_c is given in Equation 4.5.

$$Sim = \sum_{i=1}^{n} |f_{it} - f_{ic}| \tag{4.5}$$

Once the k-NN classifier determines the cases that are closest to the target case, a voting algorithm is implemented to determine the classification of the target case. We used a k-NN classifier with $k = 3$ and unanimous voting to determine the classification of the target case. Unanimons voting means that all neighbours retrieved for a target case must be spam before the target case can be classified as spam. Figure 4.4 shows that although the accuracy of the classifier using majority voting is higher than one using unanimous voting the rate of FPs is considerable lower for the classifier using unanimous voting. Since FPs are so significant in the domain of spam filtering, we chose to use unanimous voting. In addition Figures 4.4 and 4.1 both indicate that a classifier with $k = 3$ performs better than one using $k = 5$.

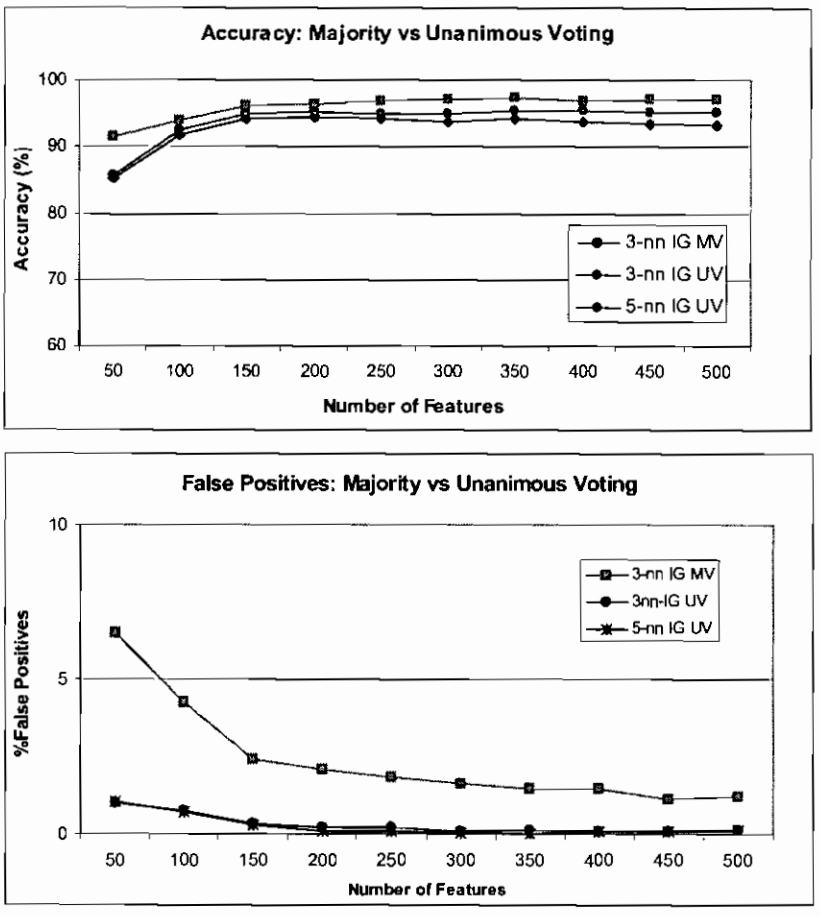


Figure 4.4: Comparison of using majority voting with unanimous voting for determining the classification of a target email. Experiments involved 10-fold cross validation on a dataset of 1000 emails, 500 spam and 500 legitimate.

Further evidence to support using unanimous voting over majority voting is shown in Figure 4.5. McNemar's test (see Section 5.1.2) was used to test whether the differences in percentage error and FP rate using unanimous voting over majority voting were significant. For datasets 1, 3 and 4 the lower FP rate was significant at 99.9%, while for dataset 2 the lower FP rate was significant at the 99% level. Looking at the effect on the accuracy of the different voting approaches, unanimous voting gives a lower error for dataset 1 with 99% significance. In the other 3 datasets the error for unanimous voting is higher than for majority voting, but this difference is only significant for dataset 4 (at 99.9% level).

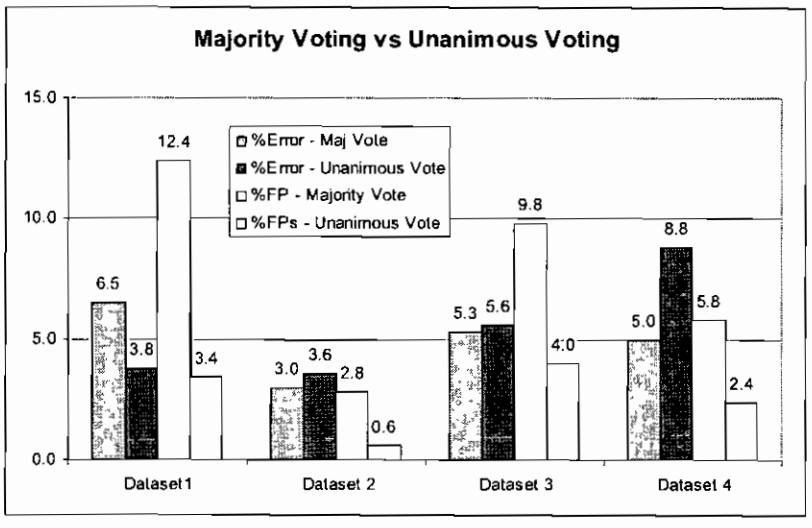


Figure 4.5: Comparison of using majority voting with unanimous voting for determining the classification of a target email. Experiments involved 10-fold cross validation on four datasets of 1000 emails, 500 spam and 500 legitimate described in Section 5.1.1. The datasets were edited (see Section 4.2 for details) and used a k-NN classifier with $k = 3$.

4.1.5 Case Retrieval

The standard k-NN algorithm individually calculates the similarity of each case in a case-base to the target case. This approach is quite inefficient in domains where there are missing features in cases. Because our spam cases have these characteristics, and our feature representation was binary, we use the alternative similarity retrieval technique, Case Retrieval Nets (CRNs) (Lenz et al. 1998a) discussed in section 3.1.2.

We implemented a CRN for case retrieval that was configurable for different k-nearest neighbour classifiers. As the features in our case representation are binary (implemented as boolean values), IEs are only included for features with a TRUE value and similarity arcs are not needed. The relevancy arcs are all weighted with a weight of one.

Figure 4.6 depicts an example of our CRN for spam filtering. Our implementation of the CRN is similar in some respects to a Concept Network Graph (CNG) (Ceglowski et al. 2003) with thresholds set so that the activations are not spread beyond the first level of nodes.

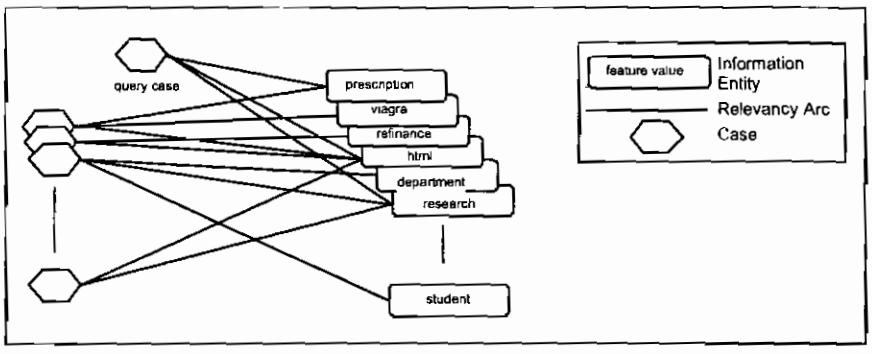


Figure 4.6: The Case Retrieval Net used in the implementation of ECUE to facilitate efficient retrieval of cases from the casebase.

4.2 Case-Base Maintenance

ECUE's case-base management strategy serves two purposes, firstly it manages the size of the training data used for training ECUE and secondly it provides the mechanism to handle the concept drift that is inherent in email, both spam and legitimate emails.

Research to date on machine learning for spam filtering has focused on static evaluations on datasets of manageable size. For instance, the LingSpam corpus (Sakkis et al. 2003, Drucker et al. 1999) contains 481 spam emails. Since a working spam filter could face this number of spam messages in a week there is a need to actively manage the training data. A key step in managing the training data is the case-base editing process that deletes noisy examples and removes redundant cases from the case-base.

Case-base editing techniques involve reducing a case-base or training set to a smaller number of cases while endeavouring to maintain or even improve the generalisation accuracy. There is significant research in this area (Smyth and Keane 1995, McKenna and Smyth 2000, Wilson and Martinez 1997, Brighton and Mellish 2002). The case-base editing technique that we used is Competence Based Editing (Delany and Cunningham 2004) which uses the competence properties of the cases in the case-base to identify noisy and redundant cases to remove.

The Competence Based Editing (CBE) technique, which is discussed in the following section, initially builds a competence model of the case-base by identifying for each case its

usefulness (represented by the cases that it contributes to classifying correctly) and also the damage that it causes (represented by the cases that it causes to be misclassified). These properties of each case are used in a two step process to identify the cases to be removed. The first stage is the competence enhancement or noise reduction stage which removes mislabelled or exceptional cases. The second stage is the competence preservation or redundancy reduction stage. Redundant cases are those that are in the centre of a cluster of cases of the same classification and are not needed for classification. The evaluation of the CBE technique is discussed in the next chapter.

The second component of our case-base maintenance policy is the procedure used to update the case-base with new examples of spam and legitimate email. The update policy operates at two levels of learning. The simplest level of learning is to simply update the case-base with new instances of spam or legitimate email. The advantage of using CBR in this first level of learning is that it requires no rebuild of the model as is necessary with other machine learning solutions. The second level of learning is to retrain the system by re-selecting features that may be more predictive of spam. This level of retraining is performed infrequently and based on newer training data. The evaluation of the effectiveness of the update policy is discussed in the next chapter.

4.3 Competence Based Editing

The objective of CBE is to remove those cases that do not contribute to the classification competence of the case-base while maintaining and even improving the generalisation accuracy of the case-base. CBE has two stages, common in most case-base editing techniques; a noise reduction stage which removes noisy and exceptional cases that can adversely effect classification competence and a redundancy reduction stage to remove superfluous cases that do not contribute to classification competence.

The noise reduction technique we present, which we call Blame-Based Noise Reduction (BBNR), extends the competence modelling ideas of Smyth and colleagues (Smyth and Keane 1995, Smyth and McKenna 1998). Their case coverage measure, used in case selection, indicates how well a case contributes to correctly classifying other cases in the case-base. We extend this model to include the notion of blame or liability. We introduce a measure for a case of how often it is the cause of, or contributes to, other cases being incorrectly classified. Traditional noise reduction mechanisms tend to focus on removing the actual cases that are misclassified. However, a misclassified case could have been

classified incorrectly due to the retrieved cases that contributed to its classification. In contrast to traditional approaches, we attempt to identify those cases causing the misclassifications and use this liability information coupled with coverage information to identify training cases we would be better off without. Our evaluation shows that, in the domain of spam-filtering, this is a better way of identifying noisy cases.

Some analysis of case-base editing techniques in the past has presented algorithms that aggressively prune the case-base at the cost of some classification accuracy (McKenna and Smyth 2000, Wilson and Martinez 1997). This is in effect a tendency to overfit the training data by finding minimal sets that cover the data. Our technique for redundancy reduction, which we call Conservative Redundancy Removal (CRR), focuses on a more conservative reduction of the case-base. It uses the competence characteristics of the case-base to identify and retain border cases.

4.3.1 Blame-Based Noise Reduction

Before discussing the actual Blame Based Noise Reduction (BBNR) algorithm itself, it is important to present Smyth and McKenna's competence model (Smyth and McKenna 1998) and our extensions to this model.

The Original Case-Base Competence Model

The original case-base competence modelling approach by Smyth and McKenna proposes two sets which model the local competence properties of a case within a casebase; the *reachability set* of a target case t is the set of all cases that can successfully classify t, and the *coverage set* of a target case t is the set of all cases that t can successfully classify. Using the case-base itself as representative of the target problem space, these sets can be estimated as shown in definitions 4.6 and 4.7.

$$\text{Reachability Set}(t \in C) = \{c \in C : \text{Classifies}(t, c)\} \tag{4.6}$$

$$\text{Coverage Set}(t \in C) = \{c \in C : \text{Classifies}(c, t)\} \tag{4.7}$$

where $\text{Classifies}(a, b)$ means that case b contributes to the correct classification of target case a. This means that target case a is successfully classified and case b is returned as a nearest neighbour of case a and has the same classification as case a.

The Extended Case-Base Competence Model

We extend the model to include an additional property; the *liability set* of a case t which is defined as the set of all cases that t causes to be misclassified or that t contributes to beiug misclassified, see definition 4.8.

$$\text{Liability Set}(t \in C) = \{c \in C : \text{Misclassifies}(c, t)\} \tag{4.8}$$

where Misclassifies(a, b) means that case b contributes in some way to the incorrect classification of target case a. In effect this means that when target case a is misclassified by the case-base, case b is returned as a neighbour of a but has a different classification to case a. For k-NN with $k = 1$, case b causes the misclassification but for $k > 1$ case b contributes to the misclassification. Case a is therefore a member of the liability set of case b.

The BBNR Algorithm

Wilson's noise reduction technique (Wilson 1972) is the noise reduction algorithm upon which the noise reduction phases of many of the existiug case-base editing techniques are based (Wilson and Martiuez 1997, Brighton and Mellish 2002, McKenna and Smyth 2000). Noisy cases can be considered as training cases that are incorrectly labelled. Wilson's technique removes cases that would be misclassified by the other cases, implying that these are incorrectly labelled and are therefore noisy cases. However, a misclassified case may not necessarily be a noisy case but could be classified incorrectly due to the retrieved cases which coutribute to its classification. Mislabelled cases which are retrieved as nearest neighbours of a target case can affect the classification of the target case. Therefore just because a case is uisclassified does not imply that it is noise and should be removed.

Our BBNR approach emphasises the cases that cause misclassifications rather than the cases that are misclassified. In effect we are not just accepting the presumption that if a case is misclassified it must be mislabelled but try to analyse the cause of the misclassification. In our policy on noise reduction we attempt to remove mislabelled cases; we also remove 'unhelpful' cases that cause misclassification, e.g. a case that represents an actual spam email but looks just like a legitimate email. The liability set captures this information. The greater the size of the liability set of a case, the more impact it has had on misclassifying other cases within the case-base.

It is however important to consider this in light of how well cases are performing;

how often they actually contribute to correct classifications. The coverage set captures this information. Our BBNR technique looks at all cases in the case-base that have contributed to misclassifications (i.e. have liability sets with at least one element). For each case c with a liability set of at least one element, if the cases in c 's coverage set can still be classified correctly without c then c can be deleted. The BBNR algorithm is described in Figure 4.7.

```
T = Training Set
/* Build case-base competence model */
for (each c in T)
   CSet(c) = Coverage Set of c
   LSet(c) = Liability Set of c
endfor
/* remove noisy cases */
TSet = T sorted in descending order of LSet(c) size and
   ascending order of CSet(c) size
c = first case in TSet
while (|LSet(c)| > 0)
   TSet = TSet - {c}
   misClassifiedFlag = false
   for (each x in CSet(c))
      if (x cannot be correctly classified by TSet)
         misClassifiedFlag = true
         break
      endif
   endfor
   if (misClassifiedFlag == true)
      TSet = TSet + {c}
   endif
   c = next case in TSet
endwhile
return TSet
```

Figure 4.7: Blame Based Noise Reduction (BBNR) Algorithm

This principle of identifying damaging cases is also there in IB3. Aha's IB3 algorithm is an algorithm more applicable for data streams and online learning in that the training set does not exist as a collection of cases before editing can be performed. The decision as to whether cases are kept in the case-base or not is made as the cases are presented.

There are a number of differences between IB3 and BBNR. Firstly, IB3 maintains the classification records during the editing process rather than using the competence of the full training set as BBNR does through use of the competence model. Secondly, the classification record maintained by BBNR is based on actual classifications, whereas that maintained by IB3 is based on possible or potential classifications. IB3 updates the classification record of all cases that could potentially be neighbours whereas BBNR

only uses the k retrieved neighbours to build its competence model. However, the most significant difference between the two algorithms is how they use case liability information. Although IB3 does collect information on the likely damage that certain cases may cause, it is not used actively to determine whether these potentially damaging cases should be removed or not. IB3 uses the classification accuracy, rather than classification error, to indicate how well a case is performing and waits for a case not to classify correctly at a satisfactory level before removing it. BBNR, on the other hand, uses the liability information available from the competence model of the case-base to decide whether these potentially damaging cases have any merit in being kept in the case-base.

4.3.2 Conservative Redundancy Reduction

The second stage in our competence-based editing technique is to remove redundant cases. Our proposed algorithm for removing redundant cases is based on identifying cases that are near class borders. The coverage set of a case captures this information. A large coverage set indicates that a case is situated in a cluster of cases of the same classification whereas a small coverage set indicates a case with few neighbours of the same classification. Cases near the class border will have small coverage sets. Cases with small coverage sets are presented first to be added to the edited set. For each case added to the edited set, the cases that this case can be used to classify (that is the cases that this case covers) are removed from the training set. This deletion policy is the same as McKenna and Smyth's coverage deletion rule (McKenna and Smyth 2000). However the order of presentation of cases differs from that used in the McKenna and Smyth algorithms. The CRR algorithm is presented in Figure 4.8.

Existing editing techniques are very aggressive in their pruning of cases. Various cross validation experiments using existing techniques (ICF, RTn and a number of McKenna and Smyth's algorithmic variations) over our four datasets produced edited case-base sizes ranging from 3.5% to 46.4% of original case-base size with the average edited size of 22%. Such aggressive reductions in case-base size can have a detrimental effect on generalisation accuracy. By adding the cases near class borders to the edited set first, rather than working in the reverse order (that is with cases that are in the centre of a large cluster of cases of the same classification), our coverage deletion rule results in a more conservative reduction of the case-base. This, as shown in Section 5.4, results in larger edited case-bases and improved generalisation accuracy.

```
T = Training Set
/* Build case-base competence model */
for (each c in T)
   CSet(c) = Coverage Set of c
endfor
/* remove redundant cases from case-base */
ESet = {}, /* Edited Set */
TSet = T sorted in ascending order of CSet(c) size
c = first case in TSet
while TSet <> {}
   ESet = ESet + {c}
   TSet = TSet - CSet{c}
   c = next case in TSet
endwhile
return ESet
```

Figure 4.8: Conservative Redundancy Reduction (CRR) Algorithm

4.4 ECUE Online Application Design

An online application was developed to allow the real time evaluation of the spam filtering functionality. This section describes the design of this system, discussing the technical architecture which allows the system to integrate with the email received by an individual and the main application processes that make up the actual filtering and learning functionality.

4.4.1 System Architecture

The architecture of the ECUE filtering application illustrated in Figure 4.9 has two components; the technical architecture and the application architecture[1]. The technical architecture is the framework that the spam filtering functionality works within. It is responsible for integrating with the user's mailbox to:

(i) identify when new mail arrives in order to allow it to be filtered and

(ii) flag when the user has identified False Positive (FP) or False Negative (FN) emails so that the spam filtering application can initiate its learning process using these emails.

It is worth noting that objective (ii) assumes that the user of the system interacts with the system at some level to identify FPs and FNs to allow the system to learn over time.

[1]The technical/application architecture terminology is that used in software engineering and integration communities

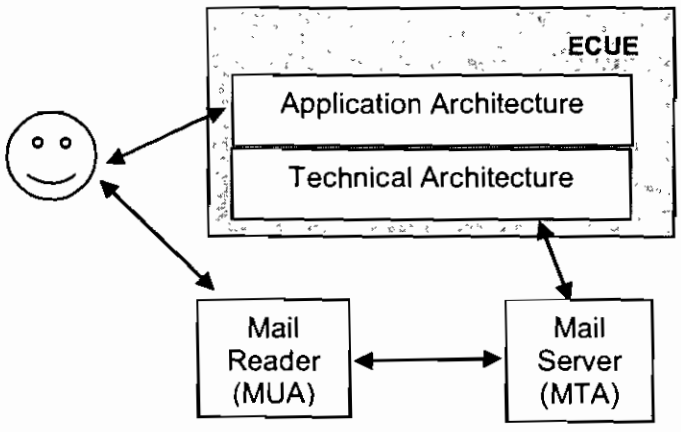


Figure 4.9: ECUE system architecture structure

The application architecture supports the actual filtering functionality. It integrates with the technical architecture in that it is notified when either a new email message needs to be filtered or the filter has made a mistake and a learning process should take place. The separation of the technical and application architecture allows independence. The architecture that supports the mailbox requirements as described above can be replaced or modified without affecting the actual filtering logic.

A key requirement of the spam filtering system is that it integrates with or works alongside the mail user agent (MUA) or mail reader rather than replacing it. This will allow the user to continue to use the mail reader software that they are familiar with. To this end, the system architecture has been designed to support initially the Internet Message Access Protocol (IMAP) protocol (Hughes 1998). The IMAP protocol is one of the two mail protocols available for receiving email messages, the other being Post Office Protocol3 (POP3). One advantage of IMAP over POP3 is that IMAP supports storing messages on the central server for access from multiple sites. By using IMAP to access the mailbox, messages can be filtered and flagged on the server and this allows the user to use any client mail reader application that supports IMAP to access and read their email. Many of the most popular mail reader applications including MS Outlook, Mozilla, Netscape and Thunderbird support IMAP. The initial prototype of the technical

architecture was tested and worked successfully on a range of MUAs and operating systems including Outlook 2003 and Mozilla Thunderbird on Windows XP Professional, Evolution 1.4 and Mozilla Thunderbird 0.5 on Fedora 1.0.

Figure 4.10 illustrates how the spam filtering system will work alongside a mail reader. Both the mail reader and the filter poll the mail transfer agent (MTA) or mail server periodically to check for new mail. As long as the polling interval (which is configurable both in mail readers and in the filter) of the filter is less than that of the mail client, most new email messages are filtered by the time the user accesses them to read them. The architecture also caters for the situation where the nser accesses an email before the filter has had a chance to filter it (see Section 4.4.4).

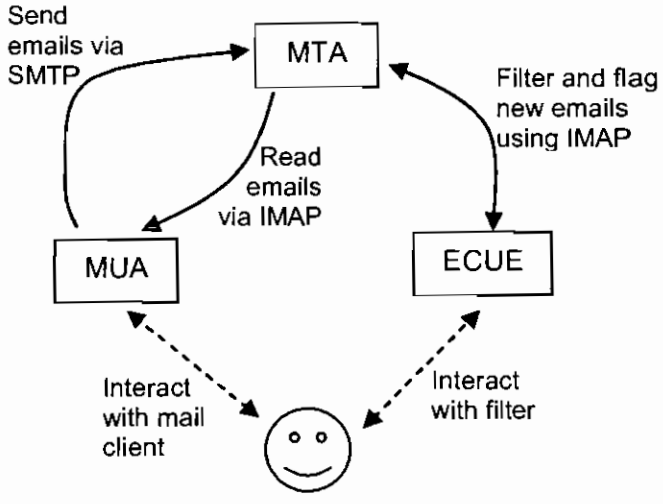


Figure 4.10: Diagram illustrating how mail client and filtering application will operate together

4.4.2 Development Platform

In order to allow the architecture to be portable across platforms the two components, the technical architecture and the filtering application were developed in Java. In addition to platform independence Java is object-oriented which facilitates the design of an architecture requiring extension and integration with other software.

4.4.3 User-System Interaction

The user and the spam filtering system have to be able to interact. This is needed for two reasons, firstly the filter has to let the nser know of emails categorised as spam and secondly the user has to be able to alert the filter to emails have been classified incorrectly. Since a requirement of the technical architecture is to integrate with existing mail readers rather than replace them, it is important to define a way of interacting with the user that is consistent across mail readers.

The system uses the IMAP mail folders as the means of system-user interaction. The filter places any emails it categorises as spam into a specific user-defined spam folder. It leaves any email that it classifies as non-spam in the Inbox. If the user finds any mails classified incorrectly they indicate this to the system by moving the emails from the folders they were in to the folder they should be in. So, an FP email should be moved from the spam folder (where the filter had placed it) into any other folder, indicating that it should not be in the spam folder. In a similar way an FN email should be moved to the spam folder to indicate that it should have been classified as spam (see Figure 4.11).

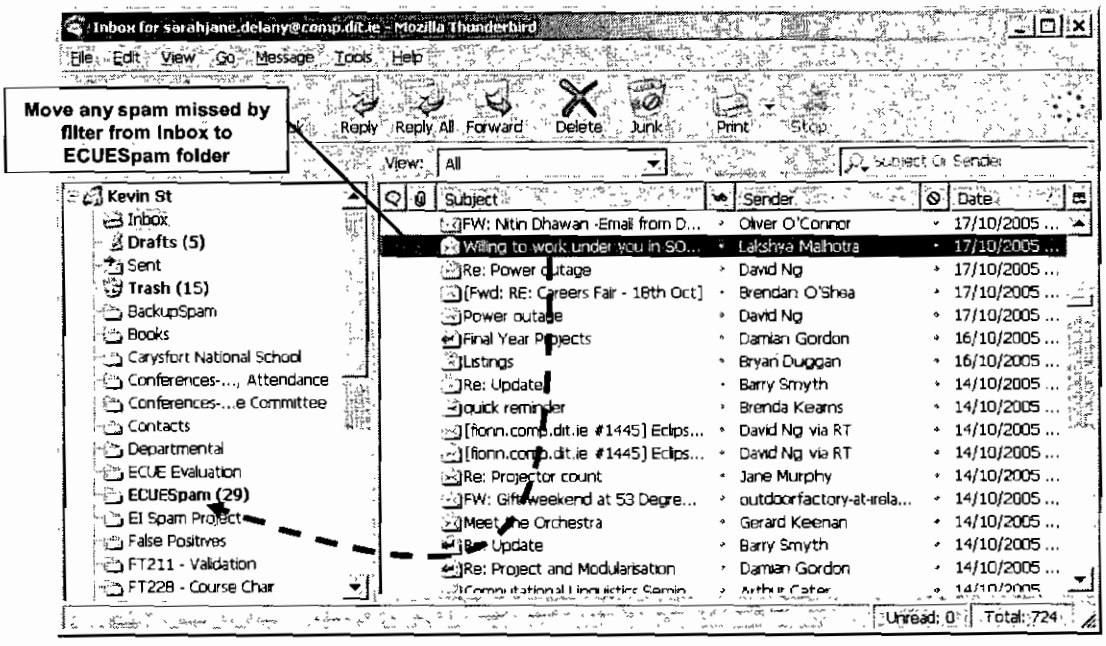


Figure 4.11: How the user interacts with the ECUE spam filter

Using this model for system-user interaction means that existing mail clients do not have to be extended to enable them to be used with the spam filtering system. All interaction is at the mail server level. This is also a familiar spam interaction model for users.

Mail folders are also used to identify the initial training data for the system. The user identifies the training emails which are to be used for the initial case-base set up by placing examples of their spam and legitimate emails into separate 'training' mail folders. These folders are identified in the configuration file and all emails in these folders are considered to be initial training data.

4.4.4 Tracking Emails

In order to track an email message as it arrives and is filtered an ECUE application specific header field is added to the email message. Email messages without the application specific field are therefore unfiltered newly arrived messages. Once an email has been filtered, a header field is added to it which identifies the email as one which has been categorised as spam or nonspam by the filter. Those emails categorised as spam are moved to the user's spam folder which will be identified at application start up time.

It is also possible, using this header field, to identify FP or FN emails. If the user finds an email in their Inbox that should have been classified as spam they move the email out of the Inbox to the spam folder. Therefore if an email with a nonspam header is found in the spam folder then this email is an FN. Similarly, if an email with a spam header is found in the Inbox, this email is an FP.

As the filter and the mail reader work alongside each other, there is a chance that the user may access a new mail before the filter has a chance to filter it. This is not serious, if the user identifies it as spam and moves it to the spam folder it will be caught by the filter as a user-filtered spam email (i.e. an email with no header in the spam folder) and updated to spam (i.e. a spam header will be added to it). If, on the other hand, a user considers the email as nonspam and leaves it where it is or moves it to a folder other than the spam folder, the filter will pick it up the next time it accesses the folder and the email will then be filtered.

Figure 4.12 depicts the state transition diagram for an email message which shows all the possible states for an email message.

4.4.5 High Level Design

Figure 4.13 is the high level class diagram of the system architecture. The non-shaded classes are the core technical architecture classes and the shaded classes allow the application to integrate with the technical architecture.

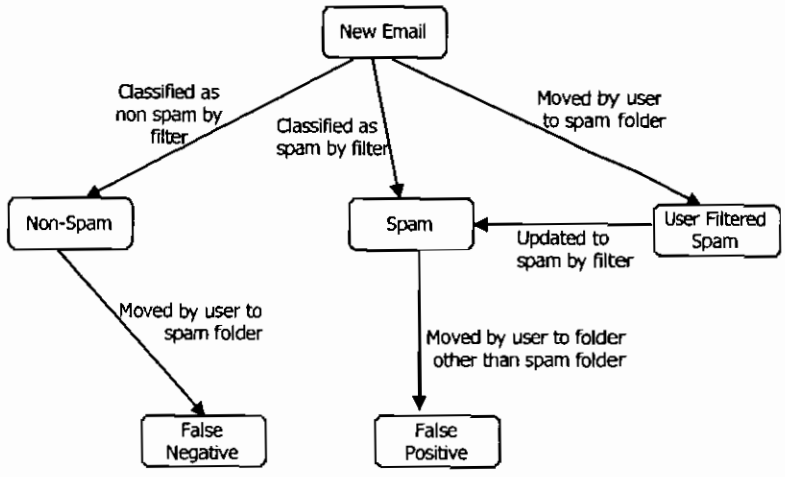


Figure 4.12: State Transition Diagram for an email message

Technical Architecture Design

The technical architecture includes two core classes, the Daemon and the Filter. The Filter is responsible for filtering email messages and determining whether they are spam or nonspam and for handling any FPs or FNs identified by the user. The Daemon manages the communication between the users mailbox on the MTA and the filter itself. Both the Daemon and the Filter run in separate threads.

The architecture has been designed in such a way that the implementation details of the Daemon or the Filter can be altered without affecting the filtering application using them. The current implementation of the daemon is that it polls the user's mailbox periodically and checks to see if the number of email messages in any folder has changed. If the number of messages has changed then either a new message has arrived to the folder or the user has moved messages between folders. In either case the daemon notifies the filter to rnn indicating which folder should be filtered.

The Filter waits for notification from the Daemon to run. The filter is activated at a folder level. It checks the headers of the emails in the folder and determines whether the email is new or whether the email is an FP or FN. If the email is new, the message is filtered, categorised as spam or non-spam and the appropriate header is added to the email. If the email is a FP or FN a report is logged (via the Reporter object) and the Learner is activated to perform the learning logic.

The MailStore class encapsulates the required functionality of user's mailbox on the IMAP server. It provides methods to connect to the mailbox and to access the mailbox

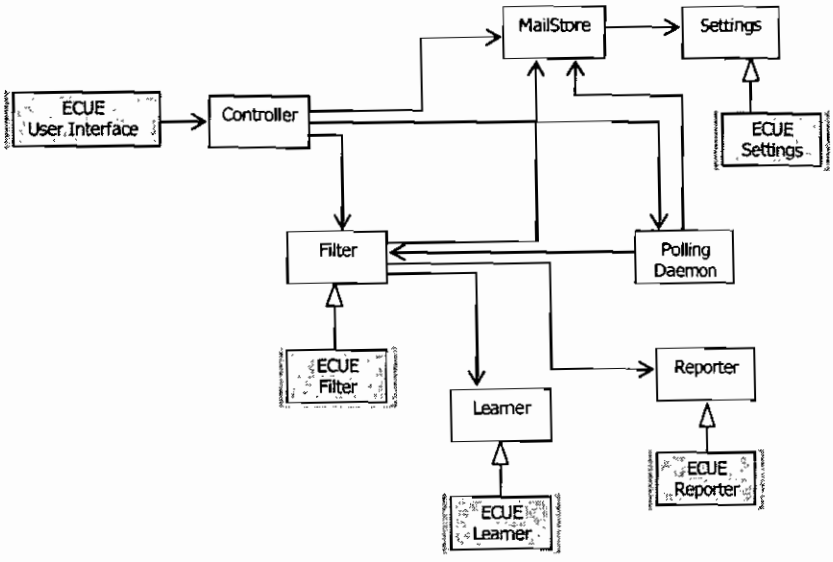


Figure 4.13: ECUE Class Diagram

folders. The Settings class encapsulates configuration settings needed for the application including details required for access to the users mailbox, e.g. host, username and password and user specific folder names e.g. the name of the folder the user has set up for spam. These configuration parameters are included in a configuration text file that is accessed and loaded at the start of the application.

The Controller class is the main control class of the application. Its main function is to start and stop the Filter and the polling Daemon. It runs in a separate thread independent of the Filter and the polling Daemon. The Learner and Reporter interfaces specify the learning and reporting processing respectively required by the spam filtering system. These interfaces are implemented in the filtering application logic.

ECUE Application Design

The application layer (as seen in Figure 4.9) provides the case-based reasoning filtering functionality. This application layer is concerned with:

(i) the set up and maintenance of a case-base of training example emails,

(ii) the classification process used to determine the classification of a new email which is filtered by the system and

(iii) the update process by which the system learns from new examples of emails.

Figure 4.14 describes the structure of the application functionality. It consists of two main processes,

(i) the SetUp process which creates a case-base from the user's emails (both spam and legitimate) and

(ii) the Filter process which performs the actual filtering of new emails and the update of the case-base with any misclassified emails.

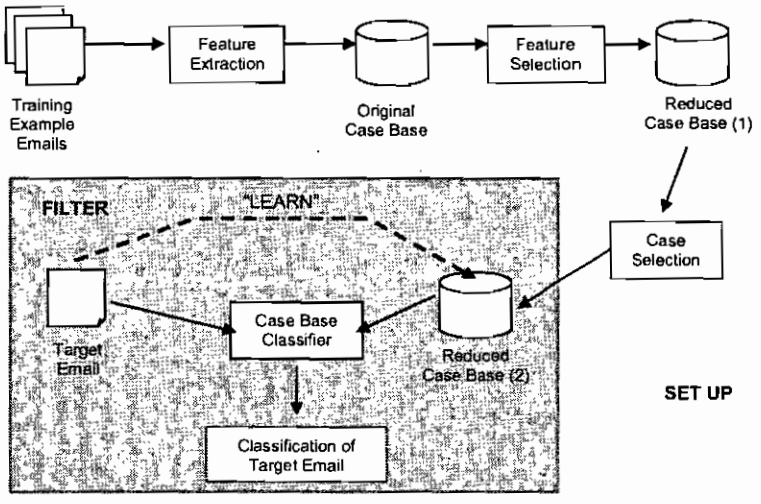


Figure 4.14: ECUE application structure

The SetUp process is represented in Figure 4.14 by the lightly shaded box whereas the actual run-time Filter process itself is represented by the dark shaded box.

Setting up a Case-base

The functionality of the case-base is encapsulated in a group of classes that manage the set-up and maintenance of the case-base and the cases that are stored within it. These classes are central to the ECUE filter and used by many of the processes described below. The main classes within this group are essentially extensions to the CBML framework (Coyle et al. 2004, Doyle et al. 2005) which is a toolkit or framework of Java classes that facilitate the set-up up and management of case-bases.

The system uses previous examples of both spam and legitimate email received by the user as training data. The first process that the emails undergo is Feature Extraction which involves parsing or tokenising the text content of the training emails into features. Three types of features were extracted; word features, letter features and statistical features as described in Section 4.1.1. The output of this process is an initial case-base of feature-value pairs for each email training example.

The feature extraction process results in a large number of features for each training email. In addition, the representation of each email will be sparse, with only a small number of the total feature set having a value greater than zero. The task of Feature Selection is to identify which of the features identified in Feature Extraction are most predictive of spam or legitimate mails. The technique used for feature selection is Information Gain as described in Section 4.1.3. The output of feature selection is a reduced feature set where each training example email has a reduced set of feature-value pairs, including only those features identified as the most predictive. The number of features used by ECUE is configurable. We used a binary representation for the features as described in Section 4.1.2.

The task of Case Selection is to apply the Competence-Based Editing technique which uses the competence properties of the examples in the case-base to remove noisy and redundant cases from the case-base. The output of this process is a reduced case-base.

The application uses different training data depending on the case-base that has to be built. If it is the first time the application is run after installation, the SetUp process uses training data that is placed by the user into two training folders in their mailbox. If a case-base is required to be built at any other time, e.g. when a featnre reselection occurs, the SetUp process uses the most recent emails received by the user as training data. In these circumstances a percentage of the training data is made up of a selection of the most recently misclassified emails. The total number of emails to be used as training data is configurable (i.e. defined in the configuration file) as is the percentage of the training data that should include the most recently misclassified emails. This percentage is made up of all the FP emails previously identified as this number will be small and an appropriate number of the previously identified FNs, randomly selected. An appropriate number of most recently correctly classified spam and non spam are then added to bring the training set up to the specified size.

Filtering and Learning

The classification of new (or target) emails is performed using the k-Nearest Neighbour algorithm as discussed in Section 4.1.4. The value of k is configurable and set up in the configuration file. The classification process uses unanimous voting to bias the classifier away from FP classifications, requiring all k neighbours retrieved by the Nearest Neighbour algorithm to be of class *spam* before the target case can be classified as spam.

When the user identifies emails that have been incorrectly classified by the system

learning should take place. There are two levels of learning built into the system:

(i) incorrectly classified emails with their appropriate classification are added to the current case-base;

(ii) a feature re-selection process and a case-base rebuild (in effect the SetUp process) is performed on more recently received emails.

This functionality is implemented in the ECUELearner class which is an application specific extension of the technical architectnre, (specifically an implementation of the Learner interface). These levels of learning supported by the system are discussed in detail in Chapter 5.

The system also provides feedback to the user via the ECUEReporter class (the implementation of the technical architecture Reporter interface) which provides statistics to the user on the performance of the filter in addition to statistics used for purposes of the evaluation.

ECUE User Interface

Figure 4.15 shows a snapshot of the screens in ECUE. The first screen is the main screen where the user can start and stop the application and initiate learning. The second screen shows the performance statistics which are displayed if the user presses the *Statistics* button. The column labelled *Overall Results* displays the overall performance since ECUE was installed. The column labelled *Session Results* show performance since the last time the filter was stopped and restarted.

Whitelisting

In order to help reduce and eliminate false positives, the system includes simple whitelisting, as discussed in Chapter 2, section 2.3.2, which operates at two levels:

(i) The user can define, in the configuration file, domains that will be acceptable to the filter. Any email that comes from these domains will be considered as legitimate.

(ii) The sender of all legitimate emails are maintained on a list and for all emails a case feature is set that indicates whether the sender is on the whitelist or not. This feature is used in the case-base retrieval process when identifying the most similar neighbours of the new email.

Database

The application uses a MySql database to store both the case-base and the user's emails in feature-value format. The structure of the database is shown in Figure 4.16.

`CBFeature` holds details of the features selected for a specific case-base. `CBCase` details the cases in a case-base while `CBCaseFeature` details the features in a specific case. `Email` and `EmailFeature` allow the details of each email that the filter classifies to be stored in feature-value format to facilitate the rebuild of the case-base. `FolderInfo` and `FoldersToFilter` hold context information about the state of the user's mailbox between runs of the filter. These entities are merely to improve performance, to allow the application to identify the folders that require filtering when the application re-starts. The `WhiteList` entity holds information about the user's whitelist.

Implementation Phases

The implementation of ECUE was phased. An initial prototype (ECUEv1) was developed and used for an initial evaluation. The functionality in this prototype included the initial case-base setup, the filtering capabilities and required the user to initiate the learning process after identifying and moving the misclassified emails between folders. ECUEv1 only included the first level of learning; updating the initial case-base with the misclassified emails.

ECUEv2 was developed after the first evaluation and additional functionality was added. The main addition to ECUEv2 was the second level of learning; a feature reselection process occurred on the user's initiation and the case-base was replaced by one built on newer examples of emails. This process is initiated when the user presses the ReSelect button (see Figure 4.15).

The maximum amount of training emails used to built the new case-base was set to 400 (up to 200 spam and 200 legitimate emails) for all evaluation users. All previously misclassified emails up to a certain percentage (20% for most users) of overall training email size were selected as training data with the most recently received emails of both classifications (spam and legitimate) bringing the training set up to the maximum specified. The new case-base was exposed to the full SetUp process as discussed above including feature selection and case-base editing.

Some minor modifications were included in ECUEv2 as a result of user feedback. The case-base update learning process was automated with all FP emails being added

immediately to the case-base. The FN emails are batched and added once a full batch has been received. The batch size is configurable but after initial trials of ECUEv2 the batch size was set to two giving almost immediate update. In addition in ECUEv1 the SetUp process was independent of the filtering process and had to be run separately. For ECUEv2 it was incorporated into the filter and initiated automatically distinguishing between the initial setup from user specified training emails and the case-base rebuild.

4.5 Conclusions

This chapter outlined the design of ECUE, the case-based system used to filter spam. Features extracted from the emails include words, characters and statistical values representing the proportion of certain types of characters in the email. We use a binary feature representation, where a word feature exists in the case if the word exists in the email. The next chapter, Chapter 5, will show why we selected a binary feature representation as the most appropriate for this domain.

A more sophisticated algorithm determines whether the character and statistical features exist in the case. This algorithm depends on the proportion of specific characters occuring in the email and compares it to the proportion that yielded the highest information gain value for that feature. The system uses a k-NN classifier with unanimous voting to reduce the FP rate as FP emails are unacceptable to most email users.

We have also introduced in this chapter a case-base editing procedure - Competence Based Editing (CBE). This case-base editing algorithm is the first stage in a two-stage case-base maintenance process which we will show can handle the concept drift inherent in both spam and legitimate emails. We also introduced the second stage of our case-base management process; the case-base update policy. The evaluation of this case-base maintenance policy will be presented and discussed in the next chapter. We will show how CBE improves generalisation accuracy and how both CBE, the case-base editing algorithm and our case-base update policy allow the handling of concept drift in legitimate email and spam.

Finally in this chapter we have outlined the technical architecture and high level design of an online system that will perform spam filtering on all emails received by an individual. The evaluation of this real time online system will also be discussed in the next chapter.

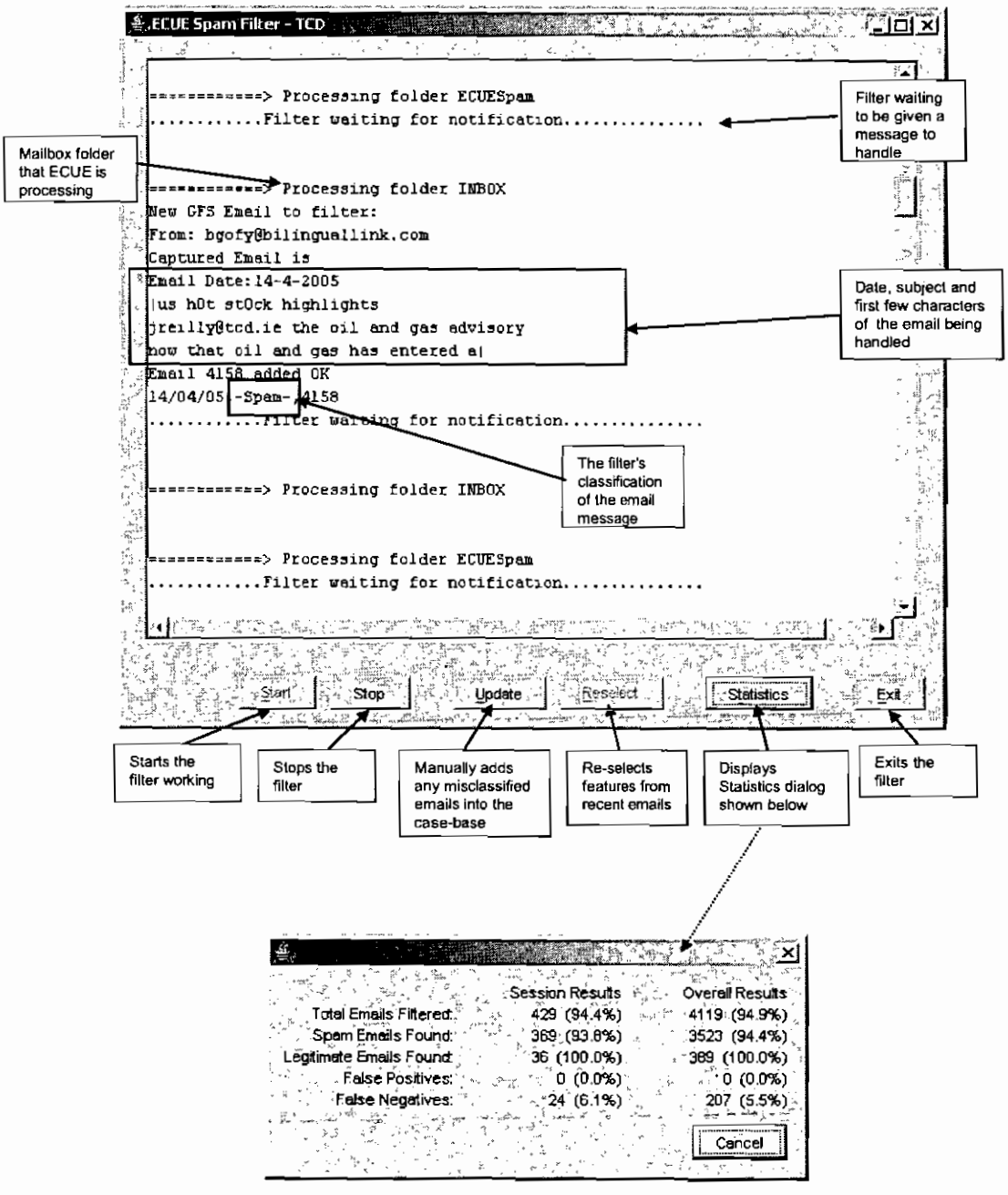


Figure 4.15: ECUE Screens

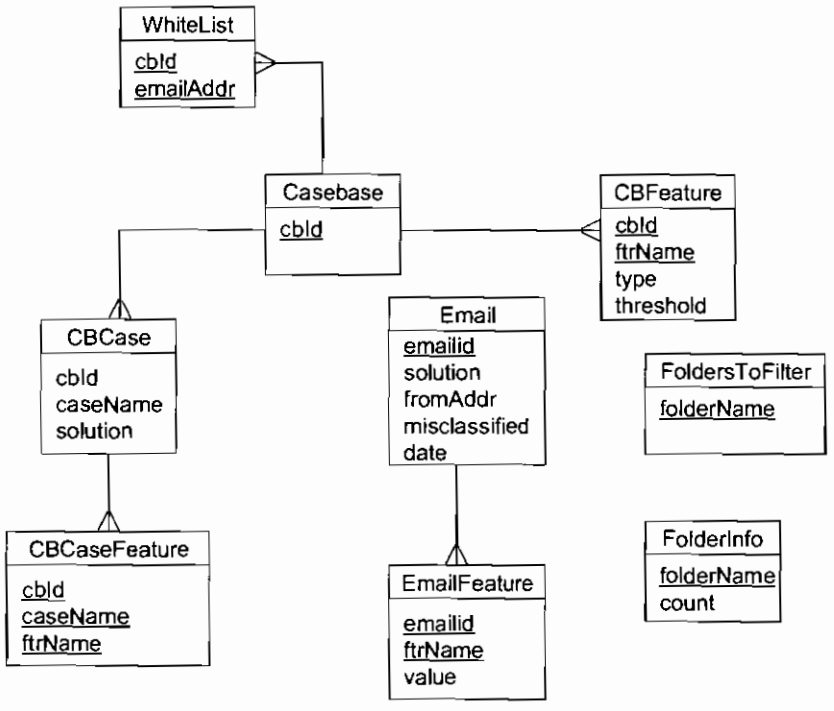


Figure 4.16: Entity Relationship Diagram for ECUE Database showing entities, relationships, primary keys and selected attributes

Chapter 5

EVALUATION of ECUE

This chapter discusses the evaluations that were performed on ECUE. We performed two types of evaluation; offline evaluations on a number of email datasets and a live real-time evaluation of the online ECUE system described in Section 4.4.

The offline evaluations discussed in this chapter support a number of conclusions. Firstly we validate our design decision of using a binary feature representation when representing a case. In addition, we show that the case-based approach that we are proposing offers advantages over the other machine learning techniques which have been applied to the problem of spam filtering. We also show how the case-base editing technique that we use increases the generalisation accuracy of an email case-base. Finally we demonstrate how the two components of our case-base maintenance policy (case-base editing and case-base update), applied at the appropriate times, can allow ECUE to track new types of spam email and legitimate email as they arrive at a user's mailbox.

The online evaluation involved installing ECUE on a number of users' machines and asking them to use it to filter their email for a period of time. A preliminary version of ECUE which included one level of learning involving updating the case-base with new examples of email, was evaluated first. This proved successful and a more sophisticated version of ECUE was then installed for these users. This version included both levels of learning, the update facility but also the periodic feature reselection. Through this real-time online evaluation we demonstrate that ECUE can recognise and handle the concept drift received by a number of users in their daily email over different periods of time.

This chapter starts with a description of the evaluation metrics used in all the evaluations in addition to describing the evaluation methodology and the datasets used in all the offline evaluations described in this chapter. It then describes the offline evaluations performed, including

(i) the determination of the most appropriate feature representation,

(ii) a comparison of case-based reasoning to other machine learning techniques,

(iii) an evaluation of the CBE technique and

(iv) an evaluation of ECUE's ability to handle concept drift.

The chapter concludes with the on-line evaluation of ECUE in a real-time live environment.

5.1 Evaluation Structure

A number of offline evaluations were performed to support the research that is discussed in this thesis. Two main types of offline evaluation were performed, static evaluations and dynamic evaluations. In general a static evaluation involves an evaluation, such as assessing the generalisation accuracy of a particular classifier or algorithm, on a number of stand-alone datasets using an n fold cross-validation approach. A dynamic evaluation is in effect an incremental validation where a very large date-ordered dataset is used and the original training data is updated as necessary with the 'test' data to allow the classifier to learn from new examples of data. The dynamic evaluation is used to assess how certain classifiers and techniques manage concept drift in a dataset over an extended period of time.

This section describes the datasets used and the rationale behind the evaluation metrics and methodology used for these experiments.

5.1.1 Datasets

All datasets for this work were derived from two corpora of spam and legitimate email collected by two individuals over a period of approximately two years up to and including December 2003 for Dataset 1 and up to and including July 2004 for Dataset 2. The legitimate emails in each corpus include a variety of personal, business and mailing list emails. The emails were ordered in date order.

For the static experiments five datasets were extracted from these corpora. Each dataset consisted of one thousand emails, five hundred spam and five hundred nonspam or legitimate emails. The emails selected were five hundred consecutive emails of each type (spam and nonspam) up to and including a particular month. The order of the emails was not important in these datasets. Two datasets were extracted from the first corpus.

Dataset 1 consisted of five hundred consecutive spam and legitimate emails received up to and including February 2003, while dataset 2's emails were the last five hundred emails of each type received up to and including November 2003. Datasets 3 and 4 were the corresponding datasets built from the second corpus up to February 2003 and November 2003 respectively while dataset 5 consisted of the last five hundred spam and legitimate emails received up to and including July 2004. Given the evolving nature of spam it was felt that these datasets gave a representative collection of spam.

For the dynamic experiments two additional large datasets (6 and 7) were extracted from the two corpora, one dataset from each corpus. These datasets included over 10,000 emails each and covered a period of one year. The emails in each of these datasets were ordered in date order. A training set of one thousand cases, five hundred spam emails and five hundred legitimate emails, was set up for each dataset. This training data included the last five hundred spam and nonspam emails received up to the end of February 2003 in the case of dataset 6 and up to the end of January 2003 in the case of dataset 7. This left the remainder of the data for testing and updating the trainiug data. Table 5.1 shows the profile of the test data across each month for both datasets 6 and 7.

Table 5.1: Profile of the testing data in datasets 6 and 7

		Feb '03	Mar	Apr	May	June	July	Aug	Sept	Oct	Nov	Dec	Jan '04	**Total**
Data Set 6	spam		629	314	216	925	917	1065	1225	1205	1830	576		**8902**
	non spam		93	228	102	89	50	71	145	103	85	105		**1076**
Data Set 7	spam	142	391	405	459	406	476	582	1849	1746	1300	954	746	**9456**
	non spam	151	56	144	234	128	19	30	182	123	113	99	130	**1409**

The class distribution of the training datasets used in the evaluations was balanced. It is not normally the case though, that individuals receive equal numbers of spam and legitimate email. A very common occurrence is for people to receive much more spam that legitimate email, although the actual quantification of "much more" can vary from one person to another. However, there are also people who receive no spam or little spam compared to the number of legitimate emails received. Given that there is no standard class distribution for spam/legitimate emails a balanced dataset was used for evaluation purposes. This is supported by Weiss and Provost (2003) who conclude that a balanced distribution is a reasonable default training distribution when the natural distribution is not available.

5.1.2 Evaluation Metrics

Since False Positive (FP) classifications (legitimate emails classified incorrectly as spam) are much more serious than False Negative (FN) classifications (spam emails classified incorrectly as legitimate), accuracy (or error) as a measure of performance does not present the full picture. Two filters with similar accuracy may have very different FP and FN rates.

In previous work on spam filtering a variety of measures have been used to report performance. The most common performance metrics are precision and recall (Gee 2003). Sakkis et al. (2003) introduce a weighted accuracy measure which incorporates a measure of how much more costly an FP is than an FN. Although these measures are useful for comparison purposes, the actual FP and FN rate are not visible so the true effectiveness of the classifier is not evident. For this reason, when reporting an error figure we will use the average within class error rate, $AvgError = (FPRate + FNRate)/2$ rather than the actual error rate ($Err = \frac{\text{number misclassified}}{\text{total emails}}$). We also report FP and FN rates separately.

In addition, for the dynamic experiments the numbers of spam and legitimate mail in the testing data sets are not equal. As the number of legitimate emails is considerably lower than spam email in these datasets, the actual error figure would follow the FN rate and not give adequate emphasis to FPs. This also supports our choice of reporting the average within class error rate over the actual error rate.

In cases where the performance of two classifiers are being compared, confidence levels are calculated using McNemar's test (Salzberg 1997, Dietterich 1998) to determine whether significant differences existed. For each test example the result is recorded and, in order to compare classifier A with B, a table such as Table 5.2 is constructed.

Table 5.2: McNemar's results table

n_{00} = the number of examples misclassified by both case-base configurations	n_{01} = the number of examples misclassified by case-base configuration A but classified correctly by B
n_{10} = the number of examples classified correctly by case-base configuration A but misclassified by B	n_{11} = the number of examples classified correctly by both case-base configurations

The total number of test examples is $n = n_{00} + n_{11} + n_{01} + n_{10}$. If no performance difference exists between the two classifiers then $n_{10} = n_{01}$. McNemar's test requires the

statistic in Equation 5.1 to be calculated.

$$\frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{5.1}$$

This statistic is distributed (approximately) as χ^2 with one degree of freedom.

The advantage that McNemars test has over the cross-validated paired t-test is a lower Type I error (the probability of incorrectly detecting a difference when no difference exists) but it also has good power (the ability to detect a difference where one exists) (Dietterich 1998).

5.1.3 Evaluation Methodology

The methodology used for the static experiments is n fold cross-validation, dividing the dataset into n stratified divisions or folds. Each fold in turn is considered as a test set with the remaining $n - 1$ folds acting as the training set. Results are accumulated over all folds and reported for the dataset.

The dynamic evaluation is an incremental evaluation. The training sets for the dynamic experiments are specified in Section 5.1.1 above. The testing emails are presented in date-received order to simulate a data stream in an online email system. In general, results are accumulated and reported at the end of each month.

A case-base was set up for each training dataset using the feature selection process described in Section 4.1.3 with 700 features in each case. The classifier used was k-nearest neighbour with $k = 3$ using unanimous voting as discussed in Section 4.1.4. Where needed, the case-base was edited with the CBE case-base editing algorithm that is discussed in Chapter 4.

5.2 Determining Feature Representation

This section outlines the evaluations that were performed to identify the most appropriate case representation for an email. This included evaluating whether to use binary or real features and whether feature weighting contributed beneficially. No domain specific feature identification was performed at this stage although work by Sahami et al. (1998) and Hidalgo et al. (2000) has indicated that the effectiveness of filters will be enhanced by their inclusion.

5.2.1 Evaluation Setup

For these experiments we used datasets 1 to 4 and 50 fold cross-validation. For each test fold and training set combination we built two case-bases, the first using a binary feature representation for the cases and the second using numeric features. Section 4.1.2 discusses how these representations were achieved. Each case-base representation was then edited using CBE (see Section 4.2). We then calculated the performance measures of the test set against each of the four case-base configurations; binary and numeric feature representation, edited and not edited.

5.2.2 Results

The results of our evaluations for each dataset and the average over all datasets are presented in Figure 5.1. It is worth noting that in the overall results, since we are calculating confidence levels using 4000 test examples, significance can be observed where the effect is quite marginal.

The results can be summarised as follows:

(i) Using numeric features on a full (non-edited) case-base has lower error rates (a significant difference at the 99% level or higher) than binary features in 3 of the 4 datasets. However, the FP performance is not significantly different except in Dataset 1 (at the 95% level).

(ii) Case-base editing improves performance for both case representations although the performance for numeric features is not as significant. (The difference for numeric features is not significant for any of the individual datasets and only measures as significant at the 95% confidence level for the overall result whereas 3 of the 4 datasets demonstrate significant improvement for binary features at the 95% level or higher with an overall difference significant at the 99.9% level).

(iii) Case-base editing also improves the performance on FPs with the binary feature representation showing higher levels of significance.

(iv) The performance of an edited case-base with binary features is not consistently significantly different from a full or edited case-base with numeric features.

We then evaluated whether feature weighting improved performance or not. Each feature was weighted with a weight equal to the IG value of the feature identified during

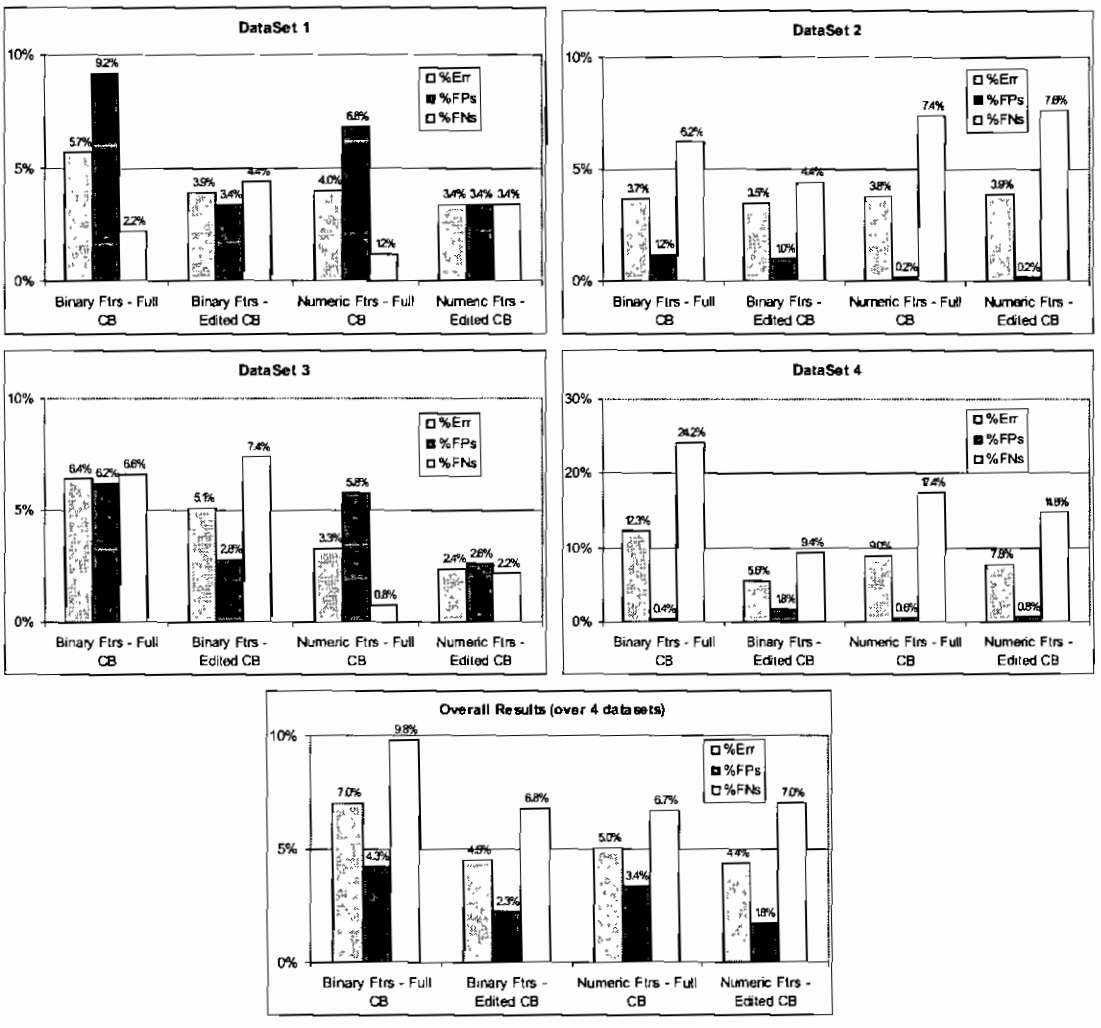


Figure 5.1: Results of evaluations of different feature representations

the feature selection process as suggested by Sakkis et al. (2003). We evaluated each case-base configuration with and without feature weighting. The results shown in Figure 5.2 can be summarised as follows:

(i) Using feature weights has a negative effect on FP performance, with 5 of the 16 comparisons showing a significant difference (at the 95% or higher) indicating that the rate of FPs is better without feature weighting. The remaining comparisons have no significant difference.

(ii) Using feature weights significantly improves the accuracy only in Dataset 2 (Nov) where 2 of the 4 comparisons show a lower error rate, with feature weighting, significant at the 99.9% or higher. The remaining datasets do not demonstrate any significant improvement using feature weighting.

Looking at the overall results for feature weighting, the best performance appears to be using numeric features on an edited case-base. There is a significant difference in accuracy (at the 95% level) using feature weighting, but no significant difference in FP rate. A close second is using binary features on an edited case-base with no feature weighting. The accuracy is not as good (a difference of 0.7%) significant at the 95% level but the difference in FP rate is not significant. Therefore in terms of classification accuracy, numeric features on an edited case-base using feature weights wins.

However, there is a considerable performance hit when using numeric features. The improvements in speed offered by the CRN are not realised for numeric features, only for binary features. While numeric features impact on response time at run-time the real performance hit comes at case-base editing time which involves classifying each case in the case-base multiple times. This is significant as it is clear that case-based editing improves accuracy. A case-base configuration that has a long response time will have a large effect on the performance of a real time system. In the case of commercial applications like spam filtering, this cannot be ignored. For these reasons we chose to use a binary feature representation and an edited case-base over the numeric features. We lose slightly in overall classification accuracy but the FP rate is not affected.

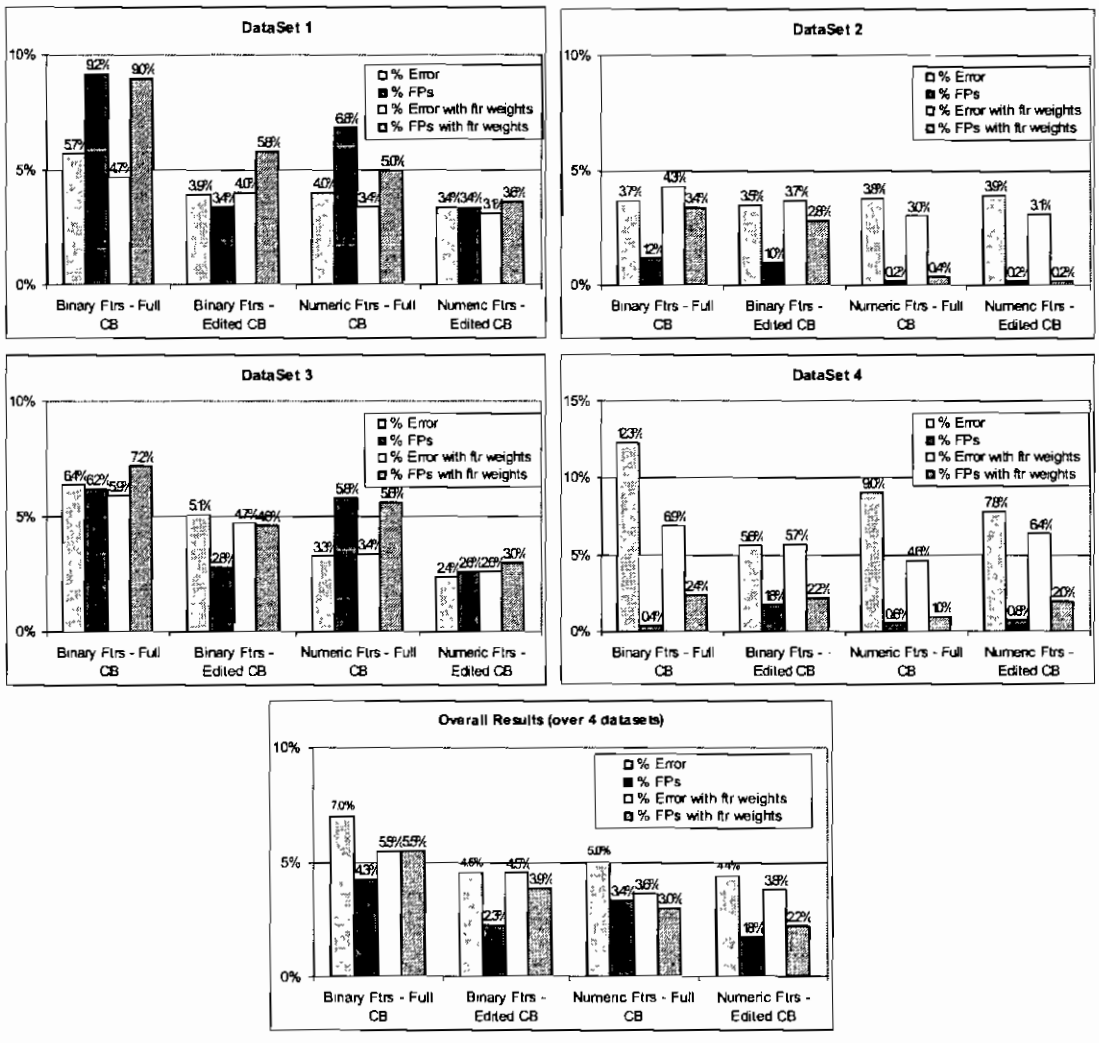


Figure 5.2: Results of feature weighting evaluations

5.3 Comparing the Case-based approach with other ML techniques

As the best configuration for the case-base has been determined, it is important to compare the case-based approach with other machine learning techniques applied to the same problem. As seen in Chapter 2 the technique of choice for commercial-based spam filtering systems is Naïve Bayes while significant academic research has been applied to using SVMs for the problem. This section compares ECUE with an NB and an SVM classifier.

Two types of evaluation were performed, a static evaluation to compare the predictive power of the different classifiers and a dynamic evaluation to see how each classifier handles the concept drift inherent in email.

5.3.1 Static Evaluation

We evaluated ECUE over the five datasets described in Section 5.1.1 using different classifiers, a k-NN classifier, an NB classifier and an SVM. The implementation of the NB implementation used is that described in Section 3.4.1 whereas the SVM implementation used was a 2-norm soft-margin SVM with a normalised dot product kernel function.

For the k-NN classifier we used a distance weighted voting algorithm to determine the classification of the target case from its k nearest neighbours. The vote returned for classification c_i for query case x_q, over the k nearest neighbours $x_1, \ldots x_k$ using distance weighted voting is given in Equation 5.2 where $\text{One}(a, b) = 1$ if $a = b$, $\text{One}(a, b) = 0$ if $a \neq b$, w_j is given in Equation 5.3, $f_i(x_j)$ is the value of feature i in case x_j and c_j is the classification of neighbour x_j. The classification with the highest vote is deemed to be the classification of the query case.

$$\text{Vote}(c_i) = \sum_{j=1}^{k} w_j \, \text{One}(c_j, c_i) \tag{5.2}$$

$$w_j = \left(\sum_{m=1}^{n} |f_m(x_q) - f_m(x_j)| \right)^2 \tag{5.3}$$

The votes for spam and non spam are normalised and the spam normalised vote is compared with a set threshold. If the spam vote is greater than the threshold the query case is considered to be spam. By varying the threshold from zero to one and plotting the resulting FP rate against one minus the FN rate an ROC curve (Bradley 1997) can be plotted. The larger the area under the ROC curve, the better the classifier.

Normalising the probabilities returned by the NB algorithm and varying the threshold for a spam classification as described above allowed an ROC curve to be plotted for the NB classifier. An ROC curve can also be graphed for the soft-margin SVM by comparing its real-valued output to a varying threshold.

Results

The results of the k-NN classifier, for an edited (labelled *edited CB*) and non edited case-base (labelled *full CB*), the NB classifier and the SVM classifier are presented in Figure 5.3. To show the detail of the curve more clearly, only the top left hand corner of the graphs are presented.

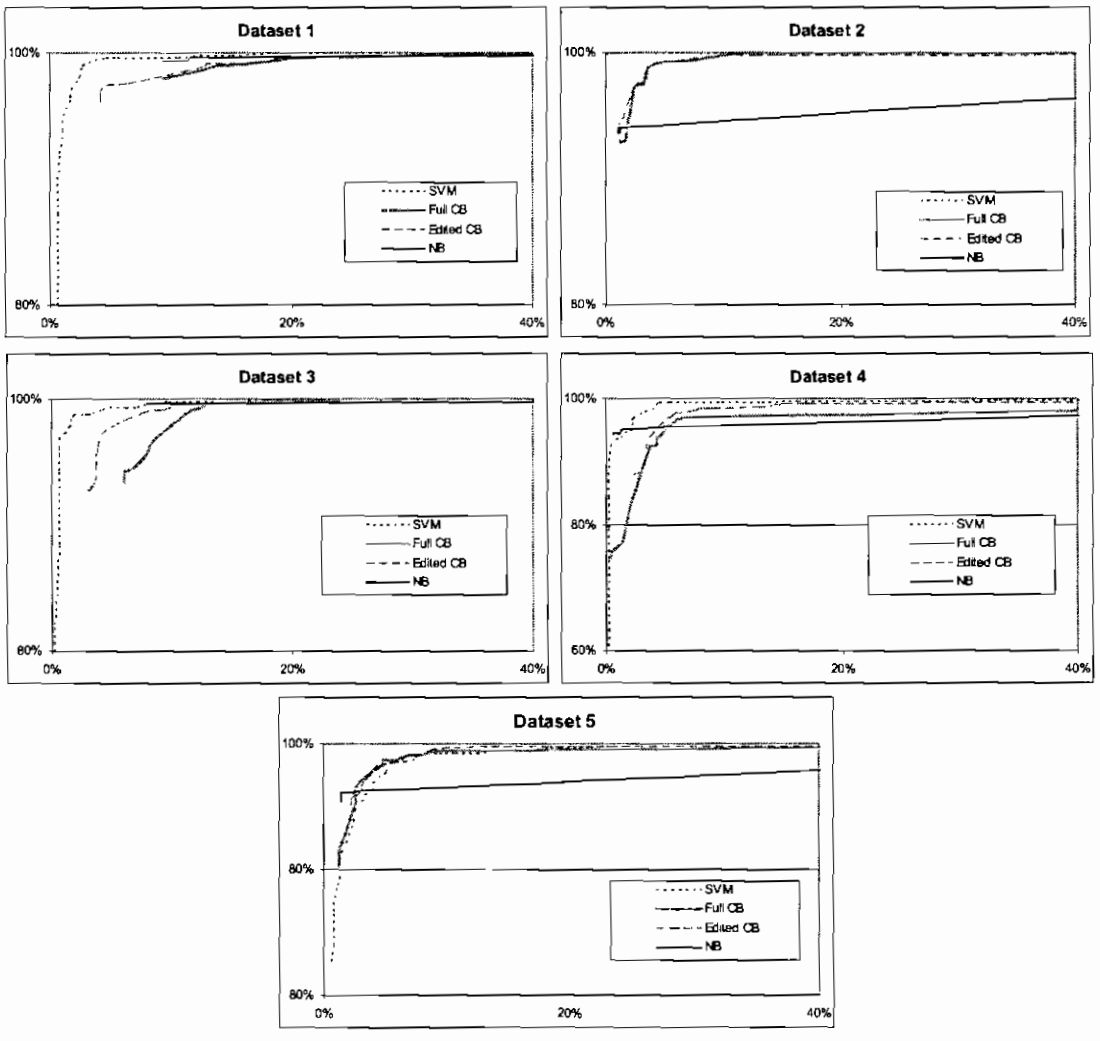


Figure 5.3: Results of comparing a k-NN classifier, an NB classifier and an SVM on static datasets

The results do not show that one classifier outperforms in all cases. SVM appears

to outperform both k-NN and NB in four out of five datasets while the k-NN classifier appears to perform best on the fifth dataset. The k-NN classifier on an edited case-base appears to perform better than the k-NN classifier on a full case-base in all cases although it is hard to distinguish between them in the case of datasets 2 and 5. The edited k-NN classifier also performs better than the NB classifier in three of the five datasets.

5.3.2 Dynamic Evaluation

We also evaluated how ECUE performs with data that is spread over a period of a year using the two datasets 6 and 7 of over 10,000 emails each, allowing the system to dynamically update its training data with examples of spam and legitimate email that were incorrectly classified.

Updating a system based on a k-NN classifier with new training data simply requires new cases to be added to the case-base. However, updating a system using NB or SVM with any new training data requires a separate learning process to recalculate the probabilities for all features and to recalculate the support vectors in SVM. Rebuilding the model for a NB classifier is relatively straight-forward, the probablilities of the features that occur in the new training examples need to be recalculated. Rebuilding the model to take a couple of new training examples into account for a SVM is computationally very expensive and not feasible for a real-time online application such as spam filtering. So, although the SVM may appear to perform well in a static evalation, the need to rebuild the SVM model for each new batch of data presents its own set of research challenges outside the scope of this research (Syed et al. 1999, Klinkenberg and Joachims 2000, Rüping 2001). Hence, the classifiers evaluated in this experiment include the k-NN and NB classifiers.

A case-base was built using the training data from each of the datasets 6 and 7 and edited nsing the CBE case-base editing algorithm. The test data was presented in date order. A number of experiments were performed, varying from making no updates to the original case-base to updating the case-base on a monthly, weekly and daily basis with those emails that were misclassified over the specified period. Our evaluation showed the best performance occurred when updating the case-base on a daily basis with any emails misclassified that day. These results are presented in Figure 5.4.

The same experiments were performed using the NB classifier on unedited training data. Due to the significance of FPs, the NB classifier was configured to be biased away from false positives by setting the threshold equal to 1.0. Figure 5.4 also includes the results of using NB.

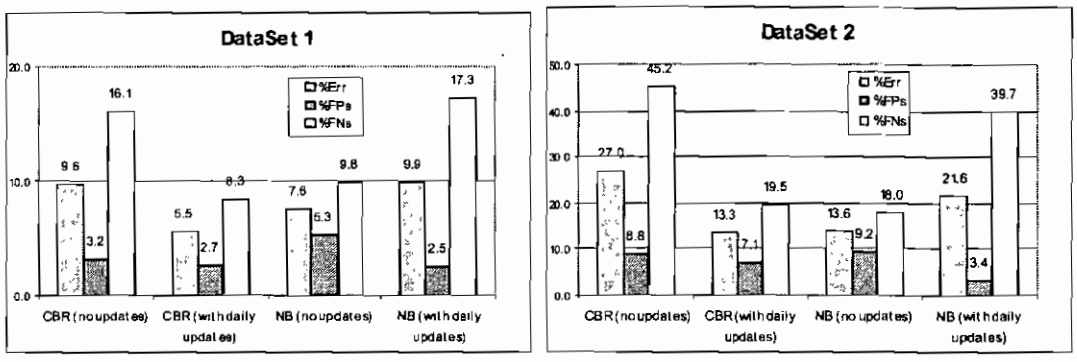


Figure 5.4: Results of preliminary evaluation of ECUE compared with a Naïve Bayes classifier over a period of time using dynamic updating

Although NB has a lower overall error rate over the datasets with no updating, the CBR system performs better in both datasets when dynamically updating the data to learn from incorrectly classified emails. It can be seen that daily updating of the training data with misclassified emails improves performance of the CBR system but has an overall detrimental effect on the NB classifier. NB with daily updates does improve the FP rate more than ECUE, possibly as the additional legitimate emails being added as training data gives better generalisation accuracy for the legitimate class, but the degradation of the FN rate has an overall negative effect on performance. CBR only needs individual marker cases to construct its model whereas NB requires a full concept description. Updating with specific cases only may affect the performance of the NB classifier however the need to train the NB classifier on a full set of data presents its own set of data management problems.

5.4 Evaluation of CBE

The evaluation of the CBE algorithm is presented at two levels; firstly, an evaluation of the performance of our competence-based BBNR algorithm against Wilson's noise reduction as used by a majority of existing case-base editing techniques and secondly, an evaluation of the performance of existing case-based editing techniques compared to our new two-phased Competence-Based Editing technique incorporating BBNR and CRR.

5.4.1 Evaluation Setup

Datasets 1 to 4, as described in Section 5.1.1, were used. The case representation used was binary (see Section 4.1.2) using the top 700 features (see Section 4.1.3) and the classifier

was k-NN with $k = 3$ and using unanimous voting (see Section 4.1.4).

The evaluation metrics reported are the average within class error, the FP rate and the FN rate, (see Section 5.1.2). For information purposes we also include the size of each case-base after editing.

For each dataset we used 20 fold cross-validation, For each test fold and training set combination we calculated the performance measures for the fnll training set without editing and the performance measures for the training set edited with each selected editing technique. Where one case-base editing technique appeared to out perform another, confidence levels were calculated using a t-test on the paired fold level results.

The Wilson's based noise reduction algorithm that we used was Repeated Edited Nearest Neighbour (RENN) which is the noise reduction algorithm used in a number of case-base editing techniques including ICF, RT3 and McKenna and Smyth's family of algorithms.

The case-base editing techniques that we evaluated include ICF, RT2, RT3 and a selection of the McKenna and Smyth's family of case-base editing techniques described in Chapter 3. The McKenna and Smyth algorithms can be identified as *adc_o*; where *a* indicates whether the addition rule is used (True/False), *d* indicates whether the deletion rule is used (T/F), *c* indicates whether the competence model is updated (T/F) and *o* indicates the order of presentation of cases. Their top two performing algorithms are FTF_*o* and FTT_*o*, where the addition rule is not used $a = F$ and the deletion rule is used $d = T$ irrespective of whether the competence model was rebuilt or not. The top two ordering sequences are order by relative coverage (RC) and reach for cover (RFC) (McKenna and Smyth 2000). Preliminary tests indicated those algorithms which require the competence model to be rebuilt after each editing step (i.e. FTT_RC and FTT_RFC) were not significantly different in accuracy but were prohibitively computationally expensive and were discarded.

5.4.2 Evaluation Results

Figure 5.5 shows the results of comparing BBNR with RENN across the 4 datasets and the overall average results across all datasets. The graphs show percentage values for error, FP and FN rates. The average size across all 20 folds of the edited casebase is indicated (on the x-axis) as a percentage of the unedited training case-base size for the individual datasets.

The results can be summarised as follows:

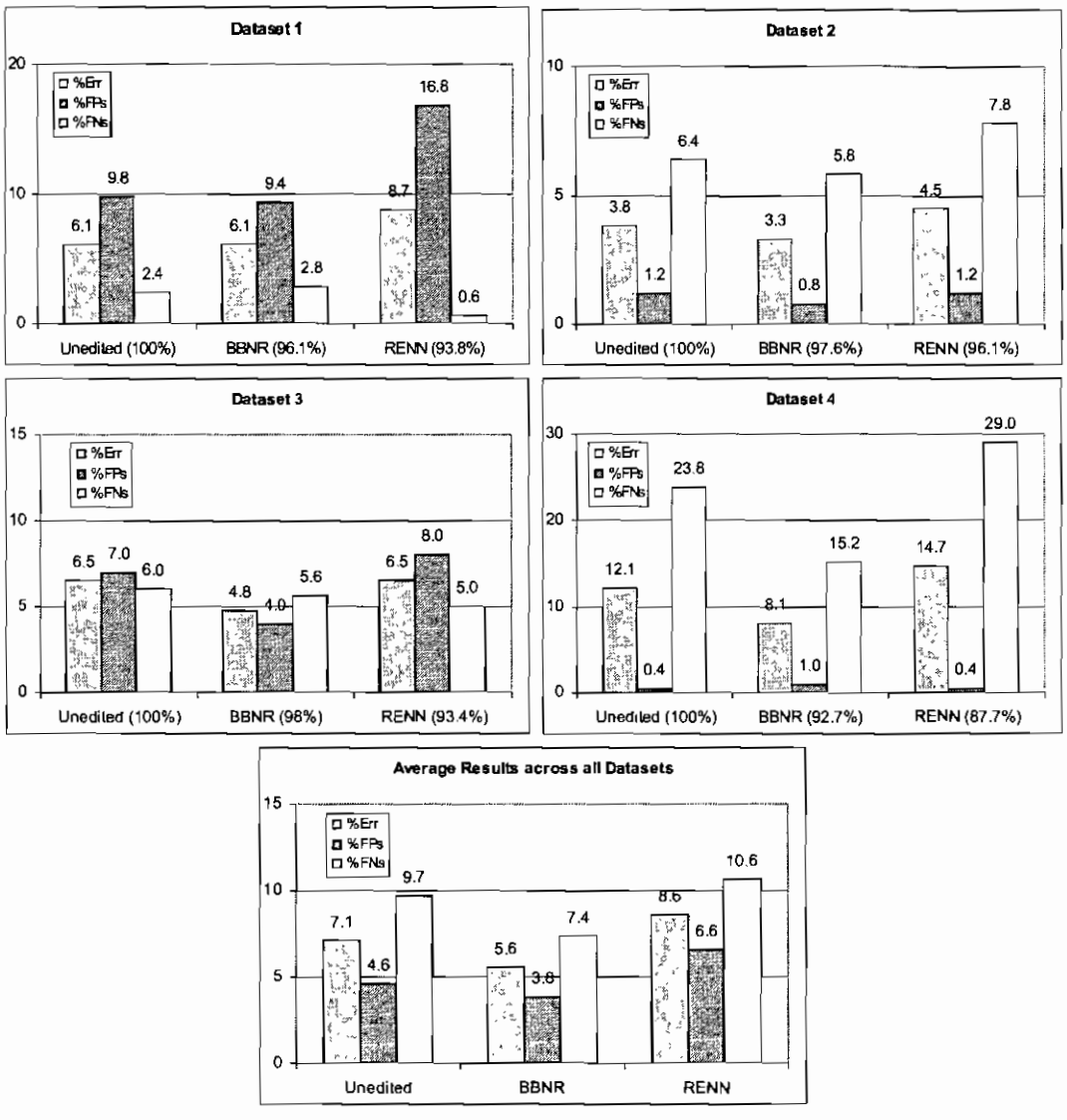


Figure 5.5: Results of BBNR compared with RENN (Wilson noise reduction)

- BBNR performs very well and has a lower error rate than RENN (significant at confidence level 99.9% across all datasets). There are also significant improvements in FP rate and FN rate (at the 99.9% level).

- The individual training sets reduced with BBNR have error rates that are at least as good as or better than the unedited training sets with the overall average showing significant improvement in FN rate and error rate at the 99.9% level and FP rate at the 99% level.

As BBNR shows better performance than Wilson noise reduction in the spam domain, we also evaluated replacing the noise reduction stage of those competence based case-base editing techniques with BBNR. Figure 5.6 displays these results for ICF, FTF_RC and FTF_RFC. Technique X with the Wilson based noise reduction phase replaced by BBNR is labelled as X-bbnr in Figure 5.6. Although RT2 and RT3 could be considered competence-based editing techniques, they use a different competence model without a liability set so BBNR was not applied to these. Figure 5.6 also includes overall average results across all datasets.

The results can be summarised as follows:

- Using BBNR to perform the noise reduction stage improves the overall performance across all the datasets for techniques ICF, FTF_RC and FTF_FRC with significant improvements in FP, FN and error rates at the 99.9% level or higher.

- Using BBNR for noise reduction in each editing technique improves performance in average error, FP and FN rates over the unedited training sets for ICF-bbnr (at levels of 95% or higher) and FTF_RFC-bbnr (at the 90% level or higher). Although FTF_RC-bbnr's FP rate shows significant improvement (at the 99.9% level) its deterioration in FN rate leads to an overall deterioration in error rate.

Figure 5.6 also includes results for RT2 and our new Competence-Based Editing (CBE) technique (i.e. BBNR+CRR). Results for RT3 were not included as RT2 outperformed RT3 for these datasets.

The results for CBE can be summarised as follows:

- Taking average results across all datasets, CBE significantly improves (at the 99.9% level) the generalisation accuracy achieved on an unedited training set of cases. The FP rate is reduced (significant at the 99.9% level) as is the FN rate (significant at the 95% level).

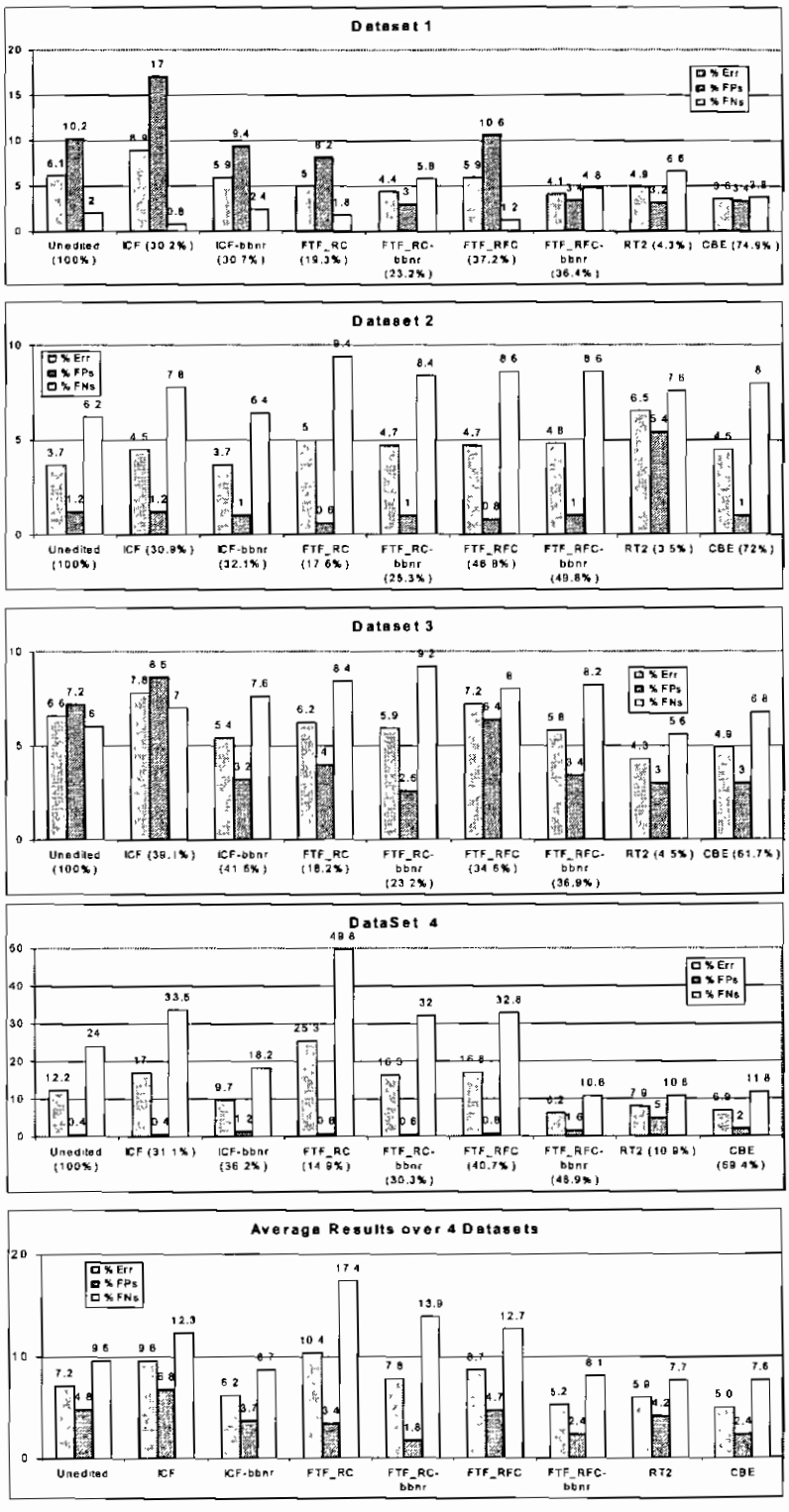


Figure 5.6: Results of various case-base editing techniques

- CBE and FTF_RFC-bbnr are the best performing editing techniques on average across all datasets with the lowest average error rates (significant at the 90% level).
- McKenna and Smyth's FTF_RFC technique with the noise reduction stage replaced by BBNR is a close second to CBE. It also demonstrates improved accuracy in average error, FP and FN rates when compared with an unedited training set, however, the improvements are at a lower level of significance.
- It may appear that CBE is out performed in specific datasets by other techniques, e.g. by RT2 in dataset 3 or ICF-bbnr in dataset 2. However CBE demonstrates the most consistent performance across all datasets.

It is interesting to note that CBE and FTF_RFC-bbnr (the top two editing techniques) result in the largest average edited casebase size (69% for CBE and 43% for FTF_RFC-bbnr).

It is worth noting at this stage that traditionally, research into case-base editing techniques tended to focus on a size versus accuracy discussion, attempting to achieve a large reduction in case-base size while not affecting, too significantly, the accuracy achieved by the reduced case-base. Improvements in technology have all but removed the size issues with respect to case-base editing. We argue that it is more important to maintain or even improve the generalisation accuracy of a case-base than to strive for large reductions in case-base size.

5.5 Case-base Update Policy

There is a natural hierarchy of learning available to a CBR system where the simplest level of learning is to simply update the case-base with new instances of spam or legitimate email. The advantage of using CBR in this first level of learning is that it requires no rebuild of the model as is necessary with other machine learning solutions. The second level of learning is to retrain the system by re-selecting features that may be more predictive of spam. This level of retraining can be performed infrequently and based on newer training data.

This section discusses the application of both these levels of learning and the evaluation of this case-base update policy. It demonstrates that an appropriate update policy will allow ECUE to learn over time from new examples of email presented as new training cases to the system.

5.5.1 Evaluation Setup

Datasets 6 and 7, as described in Section 5.1.1, were used for this evaluation. A case-base was set up for each training dataset using the feature selection process described in Section 4.1.3 with 700 features in each case. The classifier used was k-nearest neighbour with $k = 3$ using unanimous voting as discussed in Section 4.1.4. Each case-base was edited using the CBE editing algorithm described above. Each email in the testing datasets, documented in Table 5.1, was presented for classification in date-received order to closely simulate what would happen in a real-time online situation. Results were accumulated and reported at the end of each month.

Given the significance of FPs the evaluation metrics we are using here include the average error across the two classes $AvgErr = (\%FPs + \%FN)/2$ and the FP rate ($\%FP$).

5.5.2 Evaluation Results

Our results are presented at two levels. Firstly, we present an evaluation of the performance of the system at the first level of learning, i.e. simply updating the case-base with new examples of spam and legitimate email. Secondly, we evaluate the performance of the system when a model rebuild (i.e. a feature re-selection process) is carried out periodically.

The first objective was to examine at a detailed level the performance of the system with continuous updating of an edited case-base with the misclassified emails. The detailed results are presented in Figure 5.7 as *(edited cb, daily updates)*. In order to illustrate the performance improvements offered by case-base editing and continuous updating, Figure 5.7 also includes the results of the full training case-base *(full cb, no updates)* and the edited case-base *(edited cb, no updates)* when applying all the test data in date order with no updates. Finally, for comparison purposes, Figure 5.7 includes a more typical window-based updating procedure *(full cb, window-based daily updates)*. The original (unedited) case-base of 1000 cases was used and at the end of each day those cases that had been misclassified were added to the case-base. Then the window-based update procedure was achieved as follows: for each case that was added the oldest existing case of the same class was removed.

The graphs in Figure 5.7 illustrate a number of things:

(i) The concept drift in the data is evident in both datasets as can be seen by the increase in the average error over time on the graphs labelled *(full cb, no updates)*.

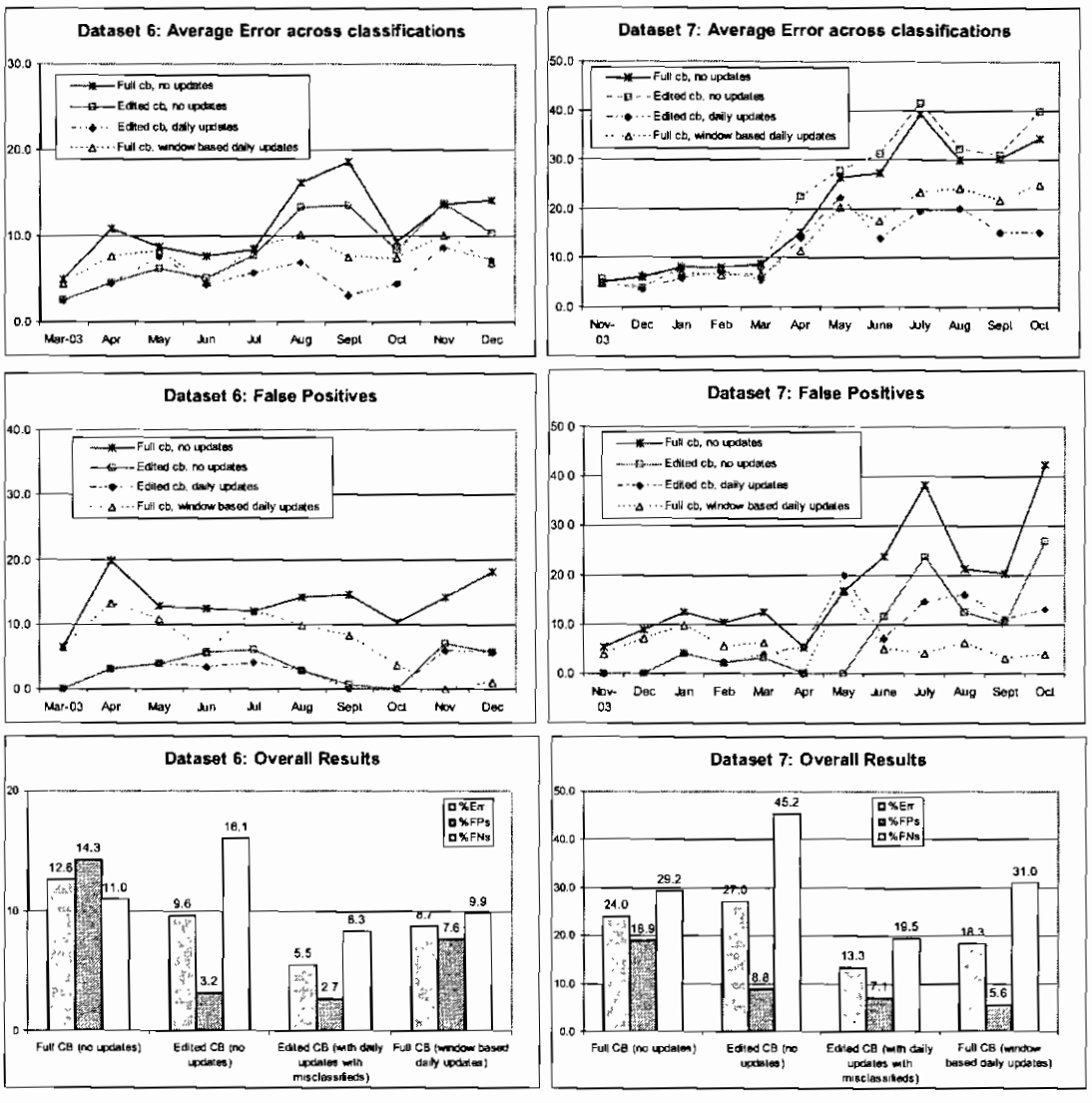


Figure 5.7: Results of applying continuous updating with misclassified emails to handle concept drift

(ii) Although case-base editing does not show a decrease in error in dataset 7, the decrease in FPs is significant, which is crucial for this domain.

(iii) Updating the edited case-base daily with misclassified cases shows a significant improvement in overall average error across classifications in both datasets. Overall FP performance is improved in both datasets too.

(iv) Although the overall FP rate is better for window-based updates than for updates to the edited case-base for dataset 7, it is considerably worse for dataset 6. The average error across classifications (taking into account the FN rate) is better for updates to an edited case-base across both datasets.

Our next experiments involved evaluating whether the next level of learning (periodically reselecting features) improved the performance gains that we had achieved by continuous updating with misclassified cases. This level of learning involves a complete model rebuild. We chose to reselect features at the end of every 3-month period. At the reselect date, a new training set of 1000 emails was constructed. This training set consisted of the last 500 spam and 500 non-spam emails received up to the reselect date. Features were selected from this training set using the same feature selection process described in Section 4.1.3. A case-base was built from this training set and was edited using CBE. Emails for the next 3 months were classified against this new case-base.

Figure 5.8 presents the results of applying feature reselection to both daily updates on an edited case-base and to window-based daily updates.

The graphs in Figure 5.8 illustrate the following results:

(i) Incorporating a periodic feature reselection and model rebuild reduces the average error across classifications for both daily update procedures.

(ii) Applying daily updates to an edited case-base and including three-monthly feature reselection results in lower error rates than using a daily window-based update procedure with feature reselect.

(iii) Although the error rates reduce in all cases with feature reselection, there is no consistent improvement in FP rate for either update technique.

An advantage of including the feature reselection and case-base editing procedure for daily updates is that it controls the size of the case-base. Table 5.3 shows the size of the case-base after each feature selection and case-editing process occurred on the two datasets.

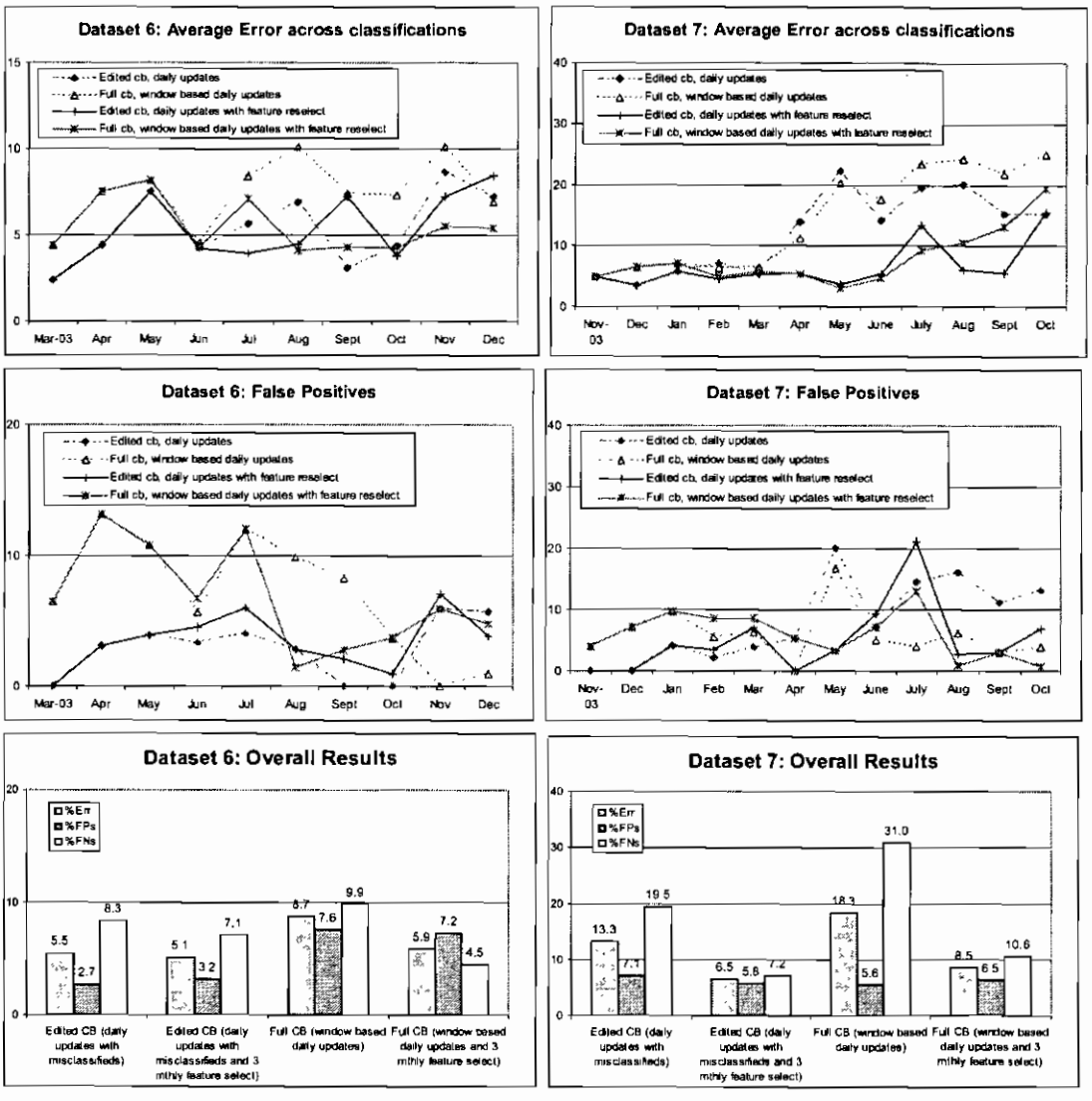


Figure 5.8: Results of applying continuous updating with misclassified emails and feature reselection to handle concept drift

In all but one situation, the edited case-base has not increased over the 3-month period beyond the size of the original case-base (1000 cases). Thus, the frequency with which feature reselection is performed can be configured to manage the size of the case-base.

Table 5.3: Case-base sizes after continuous update and 3-monthly feature reselection

Dataset 6		Dataset 7	
Size after feature reselect and edit	**Size after 3 months of updates**	**Size after feature reselect and edit**	**Size after 3 months of updates**
741	821	575	652
718	842	628	736
678	1055	661	845
719	798 (after 1 month)	580	971

5.6 Online Evaluation

ECUE was also evaluated in a real-time online setting. The following section discusses this evaluation, outlining the parameters of the evaluation including users, platforms and time duration and results. The evaluation was split into two phases. The preliminary evaluation involved an initial version of ECUE, ECUEv1, which included the case-base update facility, the first level of learning. The main online evaluation involved ECUEv2 which was enhanced to include both levels of learning, the regular update capability and the periodic feature reselection capability.

5.6.1 Evaluation Setup

The aim of these evaluations was to install the case-based spam filter (ECUE) in a 'live' environment and evaluate its performance. ECUE was designed to handle the concept drift in email. The specific objective of the evaluation was to establish whether the learning capabilities in ECUE would allow it to improve performance (i.e. identify more email correctly).

ECUE was installed on the PCs of a number of users within the Computer Science department in Trinity College Dublin (TCD) and the School of Computing in Dublin Institute of Technology (DIT). The users were lecturers, postgraduate students and researchers within the departments. Since both TCD and DIT run a gateway spam filter (SpamAssassin) each user was asked to turn off SpamAssassin and to use ECUE for filtering their email over a period of a month or so.

When users identified emails that were misclassified by ECUE, they were asked to move the emails to the appropriate mail folders. For the evaluation of ECUEv1 users were asked to initiate an update to the case-base to allow ECUE to learn from its 'mistakes' by periodically pressing the Update button. For the evaluation of ECUEv2, the case-base update facility was automated but users were asked to initiate a feature reselection process when they felt that the performance of the filter was disimproving or at least every couple of weeks.

Over the evaluation period records were maintained of how ECUE performed both with and without the learning facilities. If ECUE was learning to handle the concept drift the performance with learning should be better than the performance without learning.

5.6.2 Evaluation Results

The results of the evaluation of ECUEv1 are presented below in Table 5.4. For each user of ECUE the table lists the following:

(i) The number of days that ECUE ran on the user's PC, filtering their email.

(ii) The number of spam and legitimate emails that were filtered during that time period.

(iii) Information about the training data used, including the number of training emails used (labelled *Initial size*) and what proportion of the training data was spam email (labelled *%spam*).

(iv) The average number of days between updates (labelled *Average update (#days)*).

(v) Information about the new examples of spam and legitimate emails added to the initial training data, including the total number of new examples (labelled *#misclassifieds*) and the final size of the case-base (labelled *Final size*).

(vi) The FN rate; the proportion of spam emails that ECUE missed, i.e. those spam emails that it incorrectly classified as legitimate (labelled *%FNs*).

(vii) The FP rate; the proportion of legitimate emails that ECUE classified as spam (labelled *%FPs*).

(viii) The error rate; the overall proportion of emails that ECUE did not filter correctly (labelled *%Error*)

Table 5.4: ECUEv1 evaluation results

User		1	2	3	4	5	6
#days		28	34	30	50	70	41
Emails Filtered	#spam	969	618	80	890	3053	3248
	#legit	191	422	390	2110	747	1352
Casebase	Initial size	307	444	52	327	446	559
	%spam	49%	47%	33%	65%	49%	56%
	Final size	373	527	70	567	775	1173
	Average update (#days)	1.2	1.8	2.5	1.1	4.1	2.2
	#misclassified	66	83	18	240	329	614
% Error	No update	8.1	9.6	7.5	20.7	22.0	15.5
	With update	5.6	6.5	4.5	8.0	10.7	13.6
% FPs	No update	1.6	2.4	5.0	0.6	1.5	8.4
	With update	1.0	2.1	1.8	0.7	2.7	2.4
% FNs	No update	9.4	14.5	20.0	68.3	27.0	18.5
	With update	6.5	9.5	17.5	25.4	12.6	18.2

For all error figures (%Error, %FPs and %FNs) figures are included for how ECUE performed when it attempted to handle the concept drift (i.e. the initial training data was updated regularly with new examples to learn from), labelled *with update*, and when the ECUE simply used the initial training data in filtering and did not attempt to handle the concept drift, labelled *no update*.

An examination of the emails filtered by the users in the evaluation show that a variety of user profiles exist. Certain users receive high numbers of spam relative to the numbers of legitimate mail they receive, (user 5 and 6) whereas others (users 3 and 4) receive little spam. Different amounts of training data were used to initialise filtering for the different users. User 3 trained on very little data whereas user 6 trained on a high amount of data. Analysis of these results indicate:

(i) ECUE performed better when it updated its training data with new examples of emails in all cases. This indicates that ECUE could handle the concept drift in the email that occurred over the period of the evaluation for each user. Figures 5.9 and 5.10 show graphs of the performance of ECUE for Users 1 and 4. These graphs show the accumulated error (y-axis) over a certain number of emails (x-axis). As the user is updating the case-base manually, the *with updates* graph is quite 'spiked' reflecting that new training data is added in batches rather than continuously as in

the *without updates* graph.

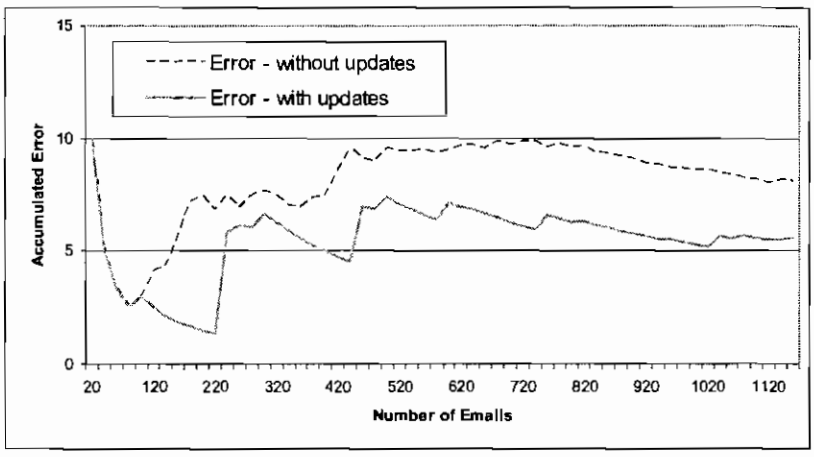


Figure 5.9: Overall Performance of ECUEv1 for User 1

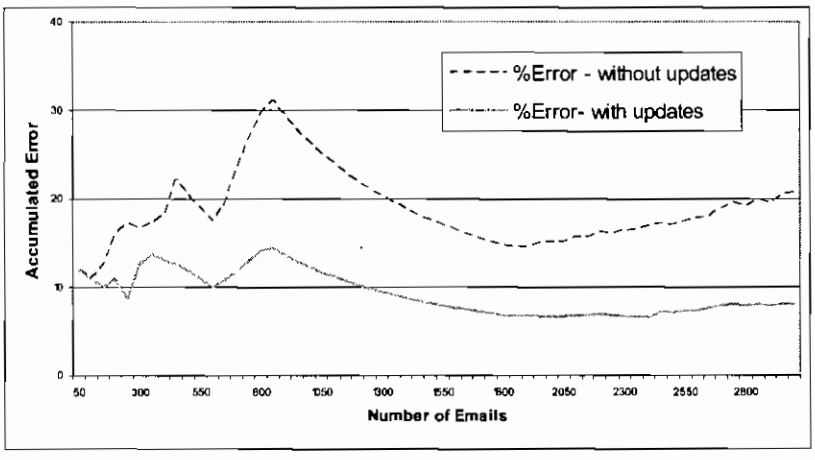


Figure 5.10: Overall Performance of ECUEv1 for User 4

(ii) ECUE learned to recognise new examples of spam over the time period as is evident from the reduction in the FN rate for all users. See Figures 5.11 and 5.12 for illustrative examples.

(iii) Improvements in the FP rate are not as consistent as the improvements in the FN rate. Four of the six users show improvements in the FP rate. Of the two remaining, one user (user 5) shows a considerable increase in FP rate from 1.5% without updating to 2.7% with updates (see Figure 5.13). This may be explained by the fact that this user received very high levels of spam and the evaluation ran for a long period (70 days). As ECUE was very successful in handling the concept drift in the spam emails (the FN rate dropping from 27% to 12.6%) the system is

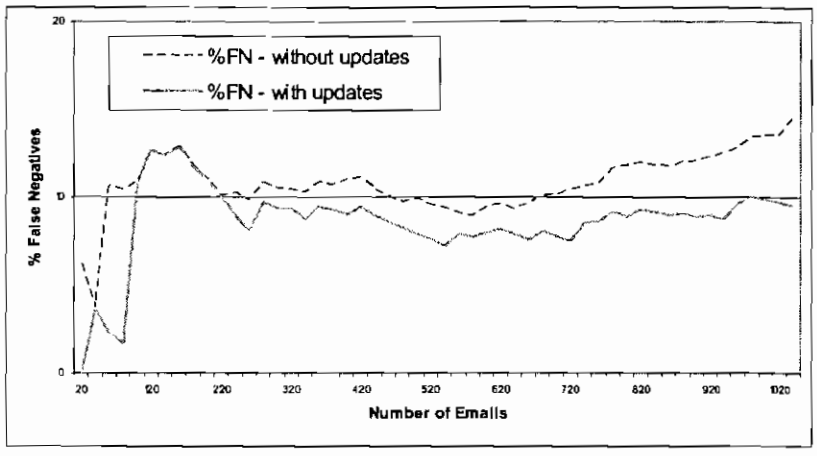


Figure 5.11: ECUEv1 FN Rate for User 2

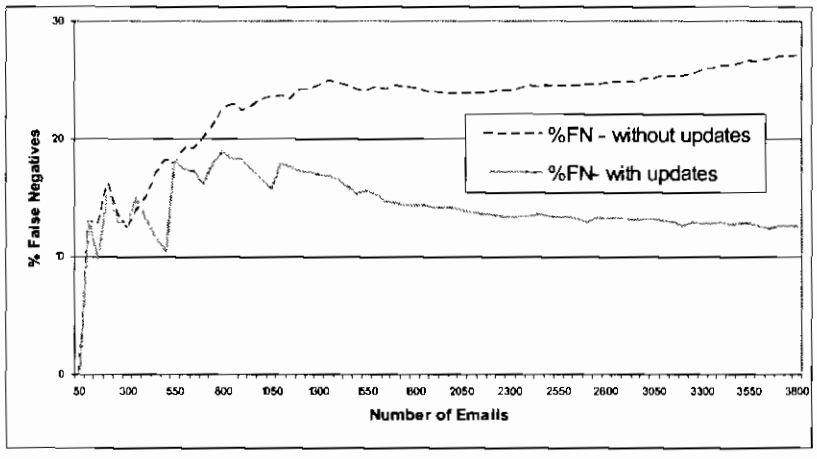


Figure 5.12: ECUEv1 FN Rate for User 5

being updated with a large number of spam emails to cope with this change and as a result becomes biased toward predicting spam.

The other user (user 4), has a very slight overall increase in FP rate. However, the FP rate for the first 2000 emails is lower with updating in place as seen in Figure 5.14.

(iv) ECUE does not perform as well for users who receive high numbers of spam emails (users 5 and 6). These users had less than 90% of their email classified correctly whereas all other users had 92% or higher classified correctly. This indicates that it is necessary to include a further level of learning to allow a feature reselection to be performed to better handle the concept drift in the spam emails.

The results of the evaluation of ECUEv2 are displayed in Table 5.5. For each user of ECUEv2 the table lists:

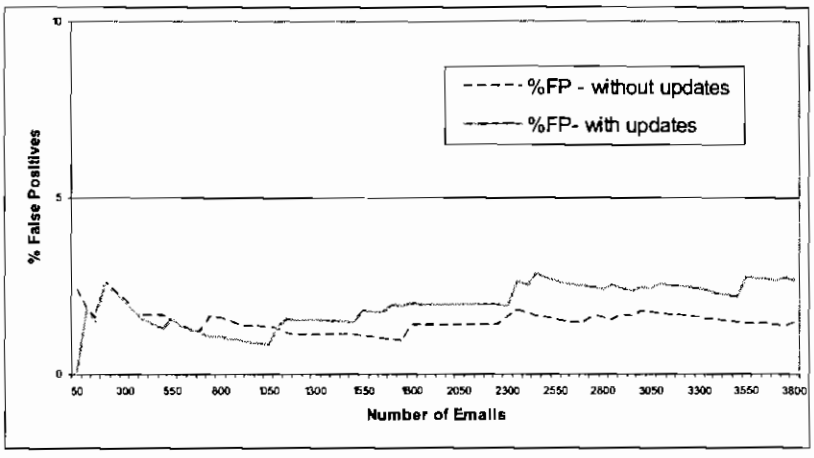


Figure 5.13: ECUEv1 FP Rate for User 5

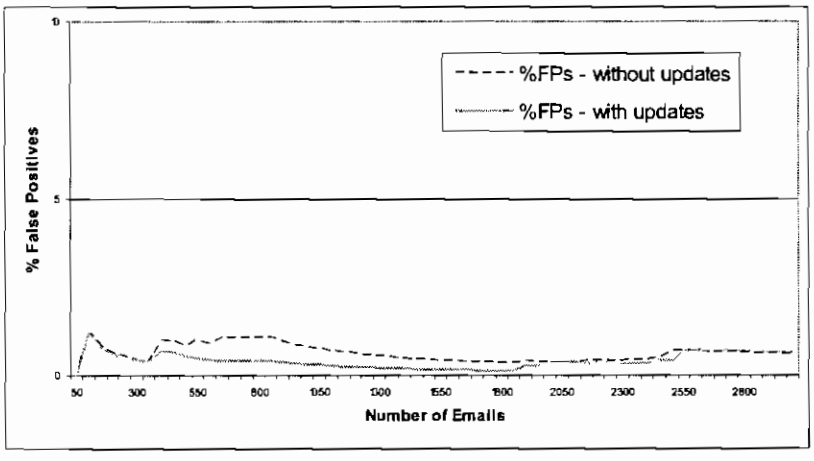


Figure 5.14: ECUEv1 FP Rate for User 4

(i) The start and end date that ECUE ran on the users PC, filtering their email.

(ii) The number of spam and legitimate emails that were filtered during that time period.

(iii) Information about the training data used, including the number of training emails used (labelled *initial size*) and what proportion of the training data was spam email (labelled *%spam*).

(iv) The number of times the user initiated the feature reselect process during the evaluation period (labelled *#feature reselects*)

(v) The FN rate; the proportion of spam emails that ECUE missed, i.e. those spam emails that it incorrectly classified as legitimate (labelled *%FNs*).

(vi) The FP rate; the proportion of legitimate emails that ECUE classified as spam

(labelled *%FPs*).

(vii) The error rate; the overall proportion of emails that ECUE did not filter correctly (labelled *%Error*)

Similarly to the first evaluation, for all error measures (%Error, %FPs and %FNs) figures are included for how ECUE performed when it attempted to handle concept drift, labelled *with update*, and when the ECUE simply used the initial training data in filtering and did not attempt to handle concept drift, labelled *no update*. Handling the concept drift at this stage involved two levels of learning; updating the initial training data regularly with new examples to learn from and performing feature reselection periodically. The arrows on all graphs related to this evaluation show when the feature re-selection process occurred. Feature reselection takes between two and three minutes depending on the size of the case-base.

Table 5.5: ECUEv2 evaluation results

User		1	2	3	4
Filter Period	**Start date**	18-11-2004	9-3-2005	20-4-2004	7-9-2005
	End date	15-07-2005	15-07-05	16-10-2005	1-11-2005
Emails Filtered	**#spam**	3689	4081	742	75
	#legit	1161	469	1480	917
Casebase	**Initial size**	308	299	201	308
	%spam	56%	54%	79%	60%
	#feature reselects	3	3	1	2
% Error	**No update**	32.8	21.8	17.3	8.1
	With update	6.1	4.7	12.1	4.3
% FPs	**No update**	0.3	0.0	1.3	7.3
	With update	0.7	0.2	1.1	0.4
% FNs	**No update**	43.2	24.4	49.3	17.3
	With update	7.8	5.2	34.0	52

Analysis of these results indicate:

(i) ECUE performed better when the update policy was used. This indicates that ECUE could handle the concept drift in the email that occurred over the period of the evaluation for each user. Figures 5.15 and 5.16 show graphs of the performance of ECUE for users 1 and 2.

(ii) ECUE learned to recognise new examples of spam over the time period as is evident

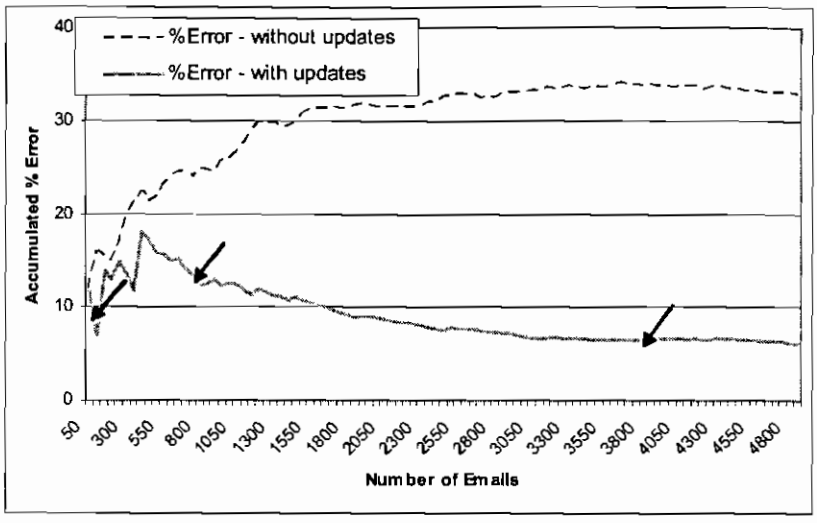


Figure 5.15: Performance of ECUEv2 for User 1

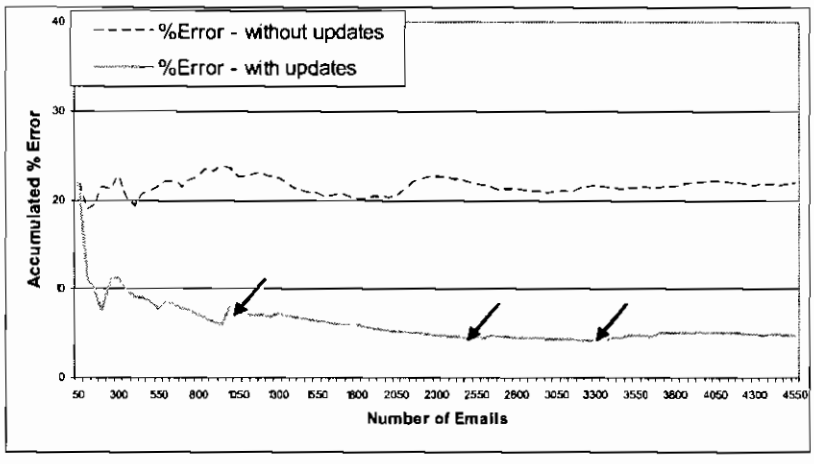


Figure 5.16: Performance of ECUEv2 for User 2

from the reduction in the FN rate for all users except user 4. Figures 5.17 and 5.18 show graphs of the FN rate for users 1 and 2 who had a significant drop in FN rate with 92% and 94% of spam respectively identified correctly.

The FN rate of user 3 did not drop as significantly as that of users 1 and 2. This may be accounted for by the fact that user 3 did not 'turn off' the gateway filter operating on his mail server. All email received by user 3 was subject to initial organisation-level spam filtering (using SpamAssassin) on the mail server before it was forwarded to user 3's personal mailbox. As such, the spam email received by user 3 was spam that had been missed by the organisation-level filter and is possibly more difficult to recognise as spam. ECUEv2 for user 3 still identified 66% of that user's spam correctly. The graph of the FN rate for user 3 is shown in Figure 5.19.

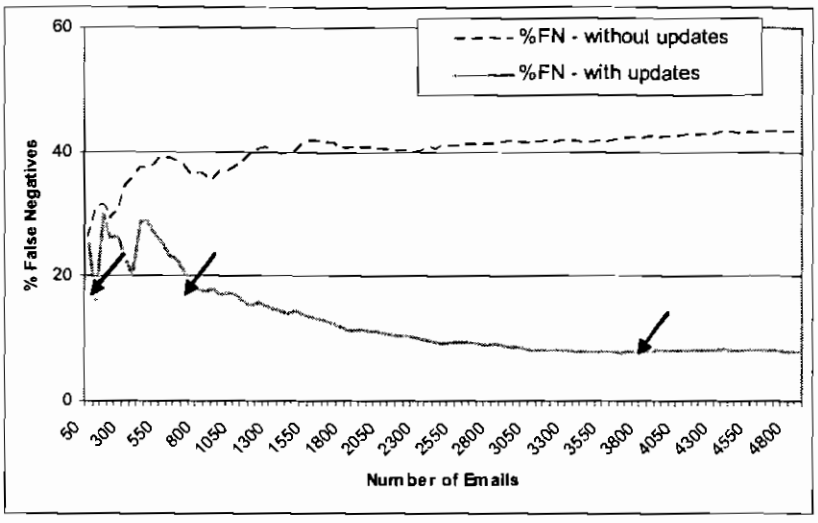


Figure 5.17: ECUEv2 FN Rate for User 1

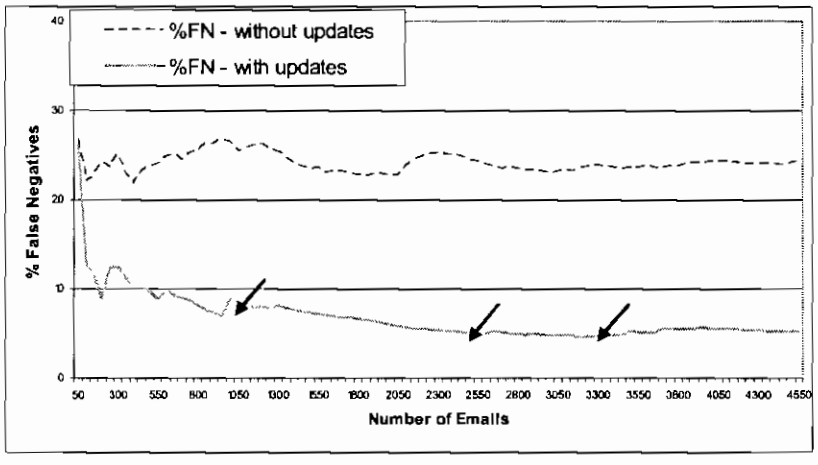


Figure 5.18: ECUEv2 FN Rate for User 2

It is also evident from this graph that it took longer for the system to stabilise, with the updated graph behaving erratically initially. This may suggest that the initial training data used to train the system was not adequately representative of the spam generally received by user 3. User 3 also only performed one feature reselect over the period of the evaluation. If more frequent reselects had been performed perhaps the FN rate would have stabilised earlier.

The FN rate of user 4 increased over the period of the evaluation. All email received by user 4 was subject to a initial organisation-level spam filter also. This organisation level filter appeared to be quite successful as only 75 spam emails were missed by the filter and passed onto the personal mailbox of user 4. On the other hand user 4 received over 900 legitimate emails, more than 12 times the amount of spam.

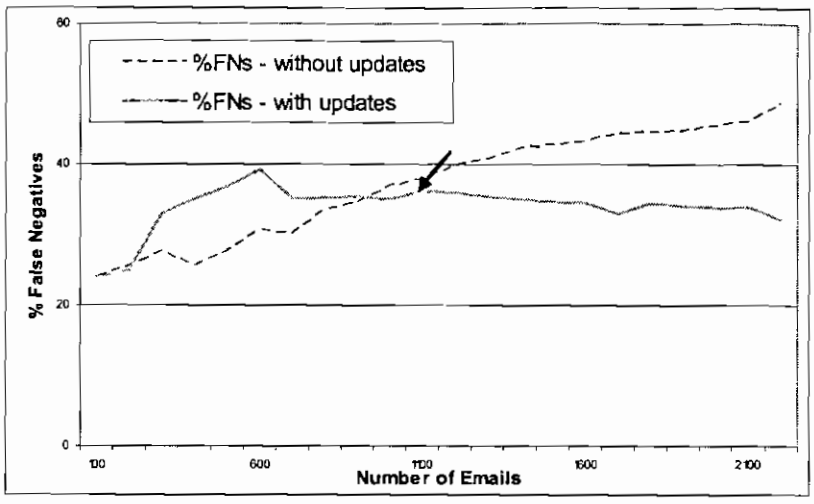


Figure 5.19: ECUEv2 FN Rate for User 3

Over the length of the evaluation the classifier becomes biased towards predicting nonspam and this may account for the increase in the FN rate for this user.

(iii) The FP rate for users 1 and 2 increased slightly over the evaluation period. The FP rate with no updates is exceedingly low for both users, 0.3% for user 1 and 0% for user 2 which is very desirable. Both of these users receive significantly more spam email that legitimate email and over time the concept of spam is changing considerably as can be seen from the graph of the rate of FNs without updates in Figures 5.17 and 5.18. The system is being updated constantly to try to cope with this change and as a result loses some accuracy in the prediction of legitimate emails.

The FP rate for user 4 decreased significantly over the evaluation period. As discussed above, user 4's filter became biased towards predicting nonspam over the period of the evaluation due the large number of legitimate mail received by this user compared with the small number of spam received by the user. The graph of this FP rate is shown in Figure 5.20.

It is clear if we examine users 2 and 4 that the distribution of their email in terms of the volume of spam and nonspam received, has influenced the bias of the classifier. Increasing the value of k in the k-NN classifier would be one way of helping to control this bias.

(iv) A shortcoming evident from the evaluation of ECUEv1 was that it did not perform as well for users who receive high numbers of spam emails. These users had less than 90% of their email classified correctly whereas all other users had 92% or higher

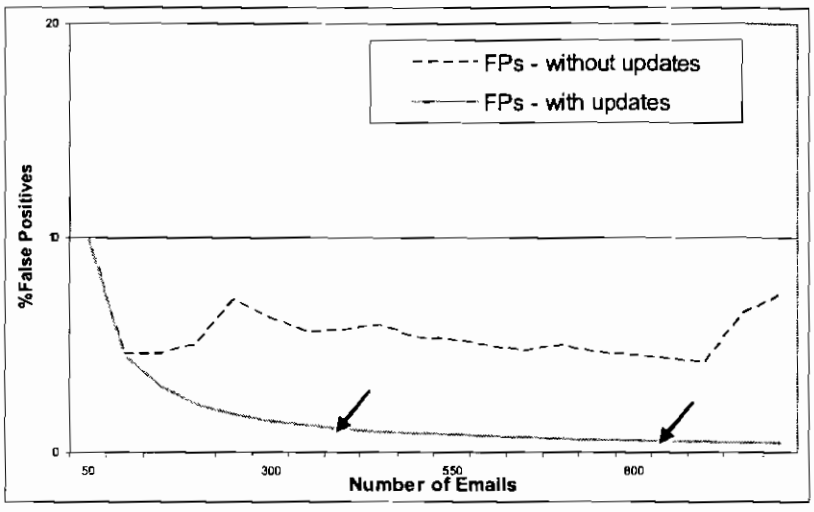


Figure 5.20: ECUEv2 FP Rate for User 4

classified correctly. Users 1 and 2 of ECUEv2 follow the expected profile of a user who receives more spam that legitimate email with the proportion of spam received by these two users varing from 76% of total mail for the User 1 to 89% of mail for the User 2. Both of these users had over 93% of their mail classified correctly. (93.9% for the User 1 and 95.3% for User 2). We can conclude from this that the additional level of learning, the periodic feature reselection process, allows the system to better handle the concept drift in the spam emails.

5.7 Conclusions

In this chapter we have discussed the evaluations performed to support our hypothesis that a lazy learner can handle concept drift in spam and legitimate email. Our approach involves a two-step case-base maintenance policy including a case-base editing technique called CBE and a case-base update policy.

We have shown how our CBE technique can assist in managing the training data and can also produce smaller case-bases with better generalisation accuracy in this domain, when compared to other popular case-base editing techniques.

Our case-base update approach entails updating the case-base at the end of each day with cases that were misclassified by the system that day and to periodically rebuild the case-base using the most recent cases from which more up-to-date features are selected. We have shown that this approach performs better than the typical window-based approach at both levels of model update.

We have also shown how the case-based approach that we are proposing offers advantages over the other machine learning techniques which have been applied to the problem of spam filtering, including the commercially popular Naïve Bayes and the recently popular Support Vector Machines. While case-based reasoning may not be a more powerful predictor evaluated using cross-validation experiments, it does offer performance benefits when dynamically updating the training data with new examples to attempt to handle concept drift.

We tested our approach in a live environment by evaluating a real-time online system that worked along side IMAP mail readers to filter the mail received at an individual's personal mailbox. These evaluations showed that ECUE was successful at filtering mail at a personal level, identifying between 92% and 94% of spam correctly with less than 1.0% false positives identified. For users who used ECUE as a second-level spam defense, operating the filter on email that had already been passed through a organisation-level gateway spam filter, ECUE still successfully filtered 48% and 66% of the spam correctly albeit with a slightly higher false positive rate for one of these users of just over 1%.

Research into concept drift which was discussed in Chapter 3 shows that ensemble approaches are among the most effective. The next chapter compares our proposed case-based approach with a variety of ensemble approaches to evaluate how ECUE performs compared to ensembles of classifiers.

Chapter 6

CASE-BASE MAINTENANCE vs. ENSEMBLES

Research into concept drift shows that ensemble approaches are among the most effective techniques for tracking concept drift (Kolter and Maloof 2003, Kuncheva 2004, Stanley 2003, Street and Kim 2001, Wang et al. 2003). Kuncheva (2004) presents the ensemble approach to learning in changing environments as online learning with forgetting. Online learning is achieved by adding new ensemble members trained with the most recent data to the ensemble and forgetting is achieved by deleting old or less-useful members. In this chapter we take three variants of this idea and compare them with our case-base maintenance approach which was discussed in Chapter 5. We also investigate the idea of dropping ensemble members that demonstrate poor performance on recent data.

This chapter starts with a discussion of the ensemble approaches for handling concept drift in spam which are used in this evaluation in Section 6.1. These approaches are evaluated in Section 6.2 and the chapter concludes in Section 6.3.

6.1 Ensemble approaches

An ensemble learner combines the results of a number of classifiers, where each base (component) classifier is constructed on a subset of the available training instances.

As discussed in Section 3.3.3, the research issues involved in using an ensemble for handling concept drift include first determining how to partition the instances into subsets with which to train the base classifiers (i.e. the data selection policy). Then a mechanism for aggregating the results of the base classifiers must be determined. Finally, a mechanism for updating the ensemble to handle new instances and 'forget' older past instances must

be established.

6.1.1 Base Classifier Structure

For the purposes of this evaluation an ensemble of classifiers $T = \{T_1, T_2, \ldots, T_n\}$ was constructed to compare against ECUE. The objective was to compare ECUE to a generic ensemble. As ECUE uses a nearest neighbour classifier, each individual ensemble member T_i was a nearest neighbour classifier built from a selection of the available training data. Each member used k-nearest neighbour (k-NN) with $k = 3$, as is used in ECUE, and a distance weighted similarity measure, see Section 5.2. Based on the accumulated similarity scores from all the k neighbours, each member T_i returns the result set $\{y_{ij} : 0 < y_{ij} < 1\}$ where y_{ij} is the score of member T_i, for classification $c_j \in C$ (C is the set of all possible classes, in our case here $C = \{spam, nonspam\}$). The y_{ij}'s are normalised such that $\sum_{j=1}^{|C|} y_{ij} = 1$.

6.1.2 Ensemble Member Data Selection

The main ensemble data selection approaches that we evaluated involve dividing the training data into blocks of fixed size organised by date and building an ensemble member using each block of training data. There are two main mechanisms used to partition the training data; a disjoint block selection mechanism which we call *Disjoint Date* and an overlapping mechanism which we call *Overlapping Date*. The Overlapping Date approach divides the training emails into overlapping sets where the percentage overlap between consecutive segments can be specified in advance. In both approaches the number of ensemble members (i.e. the number of chunks or segments) is specified.

6.1.3 Aggregation Method

The aggregation method used to determine the overall classification c_{AGG} from all ensemble members, is the classification with the largest score after a straighforward accumulation of each classification result from each ensemble member; $c_{AGG} = \text{argmax}_{j=1}^{|C|} (1/|T|) \sum_{i=1}^{|T|} y_{ij}$. This is known as Weighted Approval Voting (Plaza and Ontanon 2003). The vote for each class, *spam* and *nonspam*, is normalised such that the sum of the votes adds to 1.

By comparing the vote for the *spam* class to a threshold t where $0 < t < 1$, this aggregation method has the advantage of allowing the ensemble to be biased away from FPs. Setting a threshold $t = 0.5$ is equivalent to the weighted approval voting just described,

but setting a threshold of, e.g. $t = 0.9$, would ensure that the normalised accumulated spam vote from all member classifiers would have to be 0.9 or higher for the target email to be classified by the ensemble as spam. Setting a high value for t makes it more difficult for an email to be classified as spam, thus reducing the FPs.

A more common approach is where each member T_i returns the winning classification y_i rather than a numeric score for each classification as described above. The aggregation method in this situation is simply $c_{AGG} = \text{argmax}_{j=1}^{|C|} \sum_{i=1}^{|T|} \text{One}(y_i, c_j)$ where $\text{One}(y_i, c_j)$ returns 1 if $y_i = c_j$. Our cross validation experiments indicated that the generalisation error of this aggregation method is higher than using the numeric result of each ensemble member in the aggregation.

6.1.4 Ensemble Update Policy

The update procedure for an ensemble involved adding new members to each ensemble up to a maximum of 10 members. At that stage the oldest existing member was dropped as a new member was added, maintaining the ensemble at a maximum of 10 members. New members had equal numbers of spam and legitimate email and were added once enough new emails had been processed to get the appropriate number for a new ensemble member. A common ensemble update technique is to use a measure of global error as a trigger to create a new ensemble member. This is not appropriate in this domain as it is unacceptable to "wait" until the spam filter performs badly before adding new training data. The filter should try to pro-actively anticipate the concept drift.

As there is no standard class distribution for spam/legitimate emails a balanced case-base was used for each ensemble member which, as discussed in Section 5.1.1, is supported by Weiss and Provost (2003). Individuals normally do not receive equal numbers of spam and nonspam, so the class with the larger number of emails during that time period was randomly sampled to select training data for the new ensemble member.

Feature selection, based on Information Gain similar to the feature selection process in ECUE, was performed on each new ensemble member ensuring greater diversity between the members. It is well known that feature sub-space ensembles have good diversity (Cunningham and Carney 2000).

As ensemble pruning based on the performance of the base classifiers on recent data has been successful (Street and Kim 2001, Wang et al. 2003), we also evaluated a ensemble pruning mechanism which we call *Context Sensitive Member Selection* (CSMS) where context is defined by performance on the most recent block of training data, i.e. ensemble

members with poor accuracy on recent training examples are discarded. When a new member is to be added to the ensemble, those base classifiers that achieve a generalisation error of less than the average error across all base classifiers are dropped. The new member is always added. One of the issues with the CSMS policy is that the number of base classifiers used in the ensemble tends towards two as new members are added to the ensemble. To evaluate whether leaving more base classifiers improves the performance we used a less severe policy that removed only those base classifiers that had a generalisation error that was less than two-thirds of the difference between the best and the worst error.

6.2 Evaluation

Two types of evaluation were performed, a static evaluation which used 10-fold cross-validation and a dynamic evaluation, which was in effect an incremental validation. The static evaluation allowed an evaluation of the effectiveness or discriminating power of an ensemble in the spam filtering domain while the dynamic evaluation allowed an evaluation of how well an ensemble could handle the concept drift inherent in email.

6.2.1 Static Evaluation

The static evaluation involved comparing the generalisation accuracy of the different ensemble approaches and ECUE across datasets 1 to 4 described in Section 5.1.1.

A 10-fold cross-validation was performed on each dataset and each was evaluated (on the same cross-validation folds) using three different data selection mechanisms; Bagging, Disjoint Sets and Overlapping Sets with a 30% overlap using between 5 and 20 ensemble members. We include Bagging as a baseline technique. As it was a static cross validation, the date order of the emails was not preserved in the ensemble member generation.

As expected, Bagging had a better generalisation accuracy with a larger number of ensemble members but Disjoint Sets and Overlapping Sets had a better generalisation accuracy with ensemble members of larger size, i.e. with a smaller number of ensemble members.

The average results across the four datasets for each data selection method for the most successful ensemble size are displayed in Table 6.1. The results include figures for both the majority voting aggregation method (labelled *maj vote*) and aggregation involving a bias away from FPs with a threshold of 0.9 (labelled *with bias*) which is described in Section 6.1.3.

Table 6.1: Static evaluation of ECUE vs. ensembles

Classifier	Average over 4 datasets			
	maj vote		with bias	
	%Err	%FPs	%Err	%FPs
Disjoint Sets (5 members)	6.7%	9.1%	10.6%	1.5%
Overlapping Sets (30% overlap; 5 members)	6.7%	8.9%	10.3%	1.7%
Bagging (20 members)	6.2%	8.9%	8.1%	2.7%
ECUE	4.4%	5.8%	5.5%	2.0%

Corresponding figures are also included for ECUE, with emails presented in random order, to allow comparisons between this and the ensemble approaches. There is limited scope to bias a k-NN classifier with $k = 3$. Requiring a unanimous vote (i.e. all returned neighbours to be of classification *spam*) for spam produces the maximum bias away from FPs. Higher values of k will allow a stronger bias; however $k = 3$ produces best overall accuracy.

McNemar's test (see Section 5.1.2) was used to calculate the confidence levels between each ensemble method and ECUE to determine whether significant differences exist. The differences between each ensemble technique and ECUE were significant at the 99.9% level in all cases except for the FPs figures for the bias results where the differences were not significant in all cases.

This evaluation shows that none of the selection of ensemble approaches improves on the accuracy of ECUE in the static setting. This is not surprising and is predicted by Breiman (1996) who points out that different training datasets will not produce diversity in ensembles of nearest neighbour classifiers, thus there will be no increase in accuracy.

One benefit arising from the ensemble approach is the potential to have greater control over the level of FPs with the ensemble than with the single classifier. Setting a threshold of $t = 0.9$ on the ensembles and using unanimous voting on ECUE produces better FP figures for the ensemble approaches than for ECUE, albeit at a considerable cost in FNs and therefore accuracy. However, it is clear from comparisons of the majority voting alternatives (i.e. no bias) that the *discriminating* power of the ensembles is, if anything, worse than ECUE.

6.2.2 Dynamic Evaluation

The dynamic evaluation involved comparing the ensemble alternatives with ECUE using the datasets 6 and 7 as described in Section 5.1.1. The test emails were presented for

classification in date order.

An update policy was used to regularly update the initial classifier. The update policy used for the ECUE classifier was that described in the previous chapter. The ensemble update policy is described above in Section 6.1.4. A key difference between the static and dynamic evaluation is that the update policy attempts to introduce diversity in the ensemble by using feature subset selection in the new base classifiers added to the ensemble.

We evaluated both the Disjoint Date and the Overlapping Date ensemble data selection methods but not Bagging as it does not lend itself to an update policy to handle concept drift. Disjoint Date and Overlapping Date correspond to the Disjoint Sets and Overlapping Sets used in the static evaluation.

Figure 6.1 shows, for both datasets, how concept drift is handled by both an ensemble of case-base classifiers (based on the CSMS policy) and ECUE. Emails were classified in date order against the training data and results were accumulated and reported at the end of each month. The graph shows the results when no updates (labelled *no updates*) were applied to the initial training data and results with the appropriate update procedure in place (labelled *with updates*).

The overall results over the full datasets are also reported. The evaluation metrics used here include the average error across the two classes $AvgErr = (\%FPs + \%FN)/2$ and the FP rate ($\%FP$) to give adequate emphasis to FPs.

It is evident from the graphs that applying updates to the training data, for both types of classification process, helps to track the concept drift in the data. It is also clear from the figures that ECUE appears to handle the concept drift as well as the ensemble. Results of applying a sliding window approach, which is the most common technique for handling concept drift, are also included in Figure 6.1 and Table 6.2 for comparison purposes. In order to compare with the ECUE update procedure, the window size was 1000, 500 spam and 500 non spam and the frequency of "the slide" was monthly, i.e. at the start of each month a new training set was used.

Table 6.2 gives the overall results of the dynamic evaluation for the variety of ensemble techniques and for ECUE.

Comparisons of the majority voting alternatives (i.e. no bias) show that the ECUE performs better than any of the ensemble techniques in terms of lower average error. ECUE also has a lower FP rate than the ensemble techniques except in the case of Dataset 2 where the CSMS results in a slightly lower FP rate 6.7% for CSMS as compared with 7.2% for ECUE. The benefit evident from the static evaluation of the potential to have more

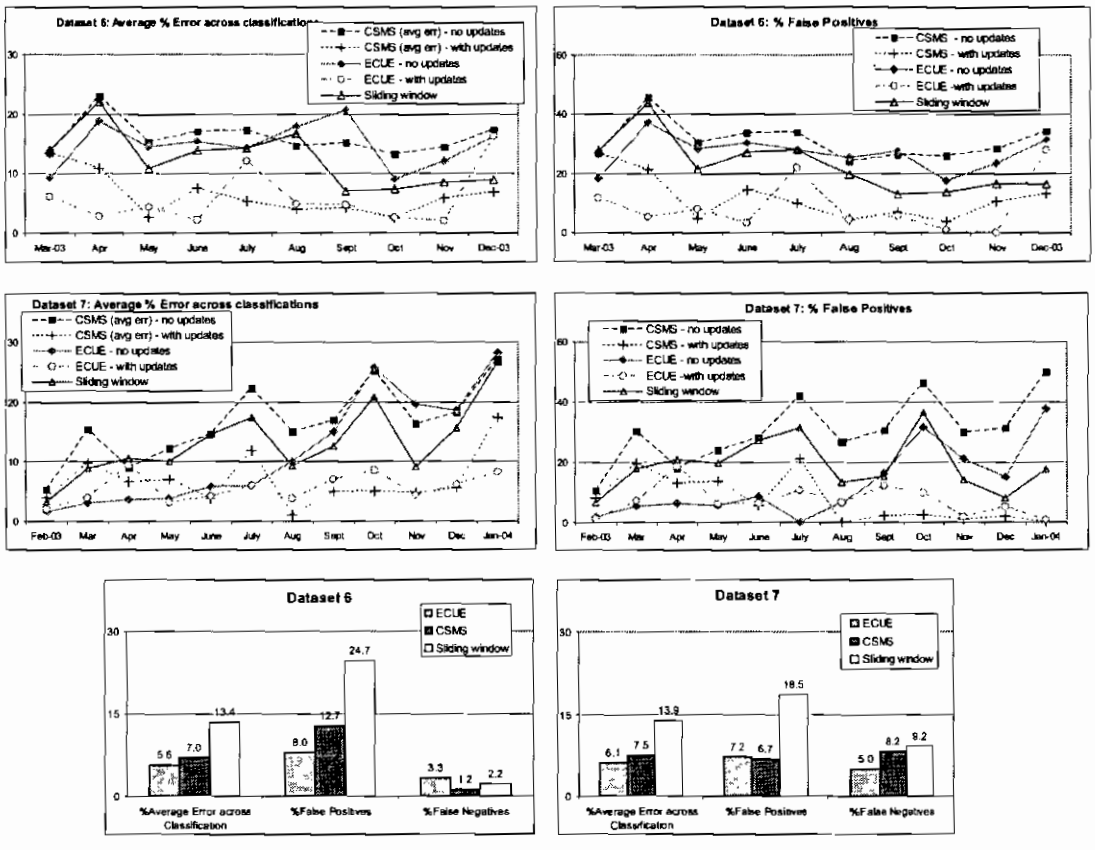


Figure 6.1: Effects of applying update policies to handle concept drift using ECUE, an ensemble of case-base classifiers and using a sliding window approach.

Table 6.2: Dynamic evaluation of ECUE vs. ensembles

Classifier	Dataset 6				Dataset 7			
	maj vote		with bias		maj vote		with bias	
	Avg %Err	%FPs	Avg %Err	%FPs	Avg %Err	%FPs	Avg %Err	%FPs
Disjoint Date	10.6	16.9	8.8	1.4	8.7	14.3	18.5	0.8
Overlapping Date (30% overlap)	10.6	16.4	7.6	2.2	9.0	10.5	20.7	0.9
CSMS (avg err)	7.0	12.7	10.0	1.2	7.5	6.7	19.2	0.6
CSMS (top 2/3)	9.4	16.2	9.3	1.5	8.3	6.9	21.0	0.6
ECUE	5.6	8.0	4.7	2.2	6.1	7.2	7.2	2.3
Sliding window	13.4	24.7	12.9	9.8	13.9	18.5	21.0	6.2

control over the level of FPs is also evident here in the dynamic evaluation. Using biasing policies described in Section 6.1.3, the FP rates for the ensemble approaches are the same or better than ECUE in all cases. However, even with these good FP rates the ensemble techniques have considerably higher average error rates than ECUE indicating a poor FN score.

The majority vote figures for the ensemble approaches show that the CSMS policy is the best performing of all the ensemble approaches. The more severe member deletion policy of removing all base classifiers less than the average error performs better than the moderate one of just removing those in the bottom third of the error range. This indicates that context sensitive selection has merit. In effect, it is removing the base classifiers that are not effective in the ensemble. However, although approaching the non-bias results for ECUE, ECUE still has lower average error indicating that it has better discriminating power.

6.3 Conclusions

This evaluation discussed in this chapter shows that the case-editing (instance selection) approach to handling concept drift is more straightforward and as effective as the ensemble alternatives we have evaluated. The discriminating power of the single classifier solution is better than that of the ensemble techniques. The most effective ensemble technique is one where the best ensemble members are selected based on an assessment of their performance on recent data. Some of the ensemble techniques return very strong results on FPs; this comes at a significant cost in overall accuracy. We have pointed out that this reflects the greater potential there is to control the bias of the classifier away from

FPs. In a sense, the strong performance of the case-editing technique is not surprising as it reflects the advantage of addressing concept drift at an instance level rather than at an ensemble member level.

The next chapter discusses how predictions of confidence can be included in a spam filtering system and outlines the advantages to the user of such a system of including estimates of prediction confidence.

Chapter 7

GENERATING CONFIDENCE ESTIMATES

ECUE has the advantage of being very effective at tracking concept drift but this requires the user to identify False Positives (FPs) and False Negatives (FNs) so that they can be used to update the case-base. Identifying FNs is not a problem because they turn up in the Inbox (i.e. spam that has been allowed through the filter). However, identifying FPs involves the user monitoring a spam folder to identify legitimate email that has been classified as spam. Our objective here is to be able to partition this class so that the user need only monitor a subset of these emails - the set for which the confidence is low.

A straightforward success criterion in this regard is the proportion of positives for which prediction confidence is high and the prediction is correct (clearly there cannot be any FPs in this set). A mechanism that could label more than 50% of the positive class (i.e. classified as spam) as confident with no FPs in this set would be useful. The lower-confidence positives could be allowed into the Inbox carrying a *Maybe-Spam* marker in the header or placed in a *Maybe-Spam* folder that would be checked periodically.

It might be expected that ranking classifiers, i.e. ones that produce numeric scores for class membership, would deliver effective estimations of prediction confidence based on thresholds on these scores. In addition to k-NN, ranking classifiers that produce numeric scores in this way include Naïve Bayes, (see Section 3.4.1), Support Vector Machines, (see Section 3.4.2), Neural Networks (Fausett 1993) and Logistic Regression (Hosmer and Lemeshow 2000). We demonstrate in this chapter that ranking classifiers such as k-NN, Naïve Bayes, SVM and Logistic Regression do not deliver effective estimates of prediction confidence as may be expected.

The chapter first discusses the basic indicators for confidence that can be used with k-

NN and we show that no single one of these measures is effective in estimating confidence. We then present some simple techniques for aggregating these basic indicators and present an evaluation on unseen data that shows a simple voting technique to be very effective.

7.1 Confidence Measures

This section describes a number of confidence measures that could be used to predict confidence in ECUE. We concentrate on using measures appropriate for a k-NN classifier. We evaluate these measures on a number of spam datasets to assess their performance at predicting confidence.

The k-NN measures that we propose to evaluate, which are described in Section 7.1.1, perform some calculation on a ranked list of neighbours of a target case. We do not use the basic classification score of the target case as ECUE uses unanimous voting in the classification process to bias the classifier away from FPs. Unanimous voting requires all the k nearest neighbours retrieved to be of classification *spam* in order for the target case to be classified as *spam*. Therefore there is no classification 'score', as such.

7.1.1 Proposed k-NN Confidence Measures

The objective of the k-NN measures is to identify those cases that are 'close' (i.e. with high similarity) to cases of the same class as the target case and are 'far' (i.e. low similarity) from cases of a different class. The closer a target case is to cases of a different class, the higher the chance that the target case is lying near or at the decision surface. Whereas the closer a case is to other cases of the same class, the higher the likelihood that it is further from the decision surface.

Similarity is determined by comparing the features extracted from the body of the email and certain header fields including the subject, the 'from' address and addresses in the 'to' and 'cc' header fields.

For each k-NN confidence measure discussed in this section the same process occurs. Each target case is classified by ECUE as either spam or nonspam. For those target cases predicted to be spam a ranked list of neighbours of the target case is retrieved. This list of neighbours is a list of all the cases in the case-base ordered by distance from the target case. Those cases with classification equal to that of the target case (i.e. with classification spam) are considered to be *like* cases, while those cases with classification of nonspam are considered to be *unlike* cases. The measures can use

- the distance between a case and its nearest neighbours (let $NN_i(t)$ denote the ith nearest neighbour of case t) or,
- the distance between the target case t and its nearest like neighbours (let $NLN_i(t)$ denote the ith nearest *like* neighbour to case t) and/or
- the distance between a case and its nearest unlike neighbours (let $NUN_i(t)$ denote the ith nearest *unlike* neighbour to case t).

The number of neighbours used in each measure is adjustable and is independent of the number of neighbours used in the initial classification. All measures are constructed to produce a high score to indicate high confidence and a low score to indicate low confidence.

Avg NUN Index

The Average Nearest Unlike Neighbour Index (Avg NUN Index) is a measure of how close the first k NUNs are to the target case t as given in Equation 7.1.

$$AvgNUNIndex(t,k) = \frac{\sum_{i=1}^{k} IndexOfNUN_i(t)}{k} \tag{7.1}$$

where $IndexOfNUN_i(t)$ is the index of the ith nearest unlike neighbour of target case t, the index being the ordinal ranking of the case in the list of NNs.

This is illustrated in Figure 7.1 where NLNs are represented by circles, NUNs are represented by stars and target cases are represented by triangles. For $k = 1$, the index of the first NUN to target case T_1 is 5 whereas the index of the first NUN to target case T_2 is 2, indicating higher confidence in the classification of T_1 than T_2.

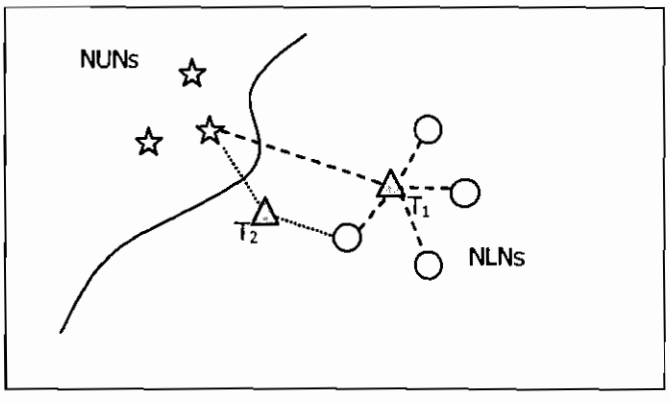


Figure 7.1: Average NUN Index Confidence Measure

Similarity Ratio

The Similarity Ratio measure calculates the ratio of the similarity between the target case t and its k NLNs to the similarity between the target case and its k NUNs, as given in Equation 7.2.

$$SimRatio(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NLN_i(t))}{\sum_{i=1}^{k} Sim(t, NUN_i(t))} \tag{7.2}$$

where $Sim(a,b)$ is the calculated similarity between cases a and b.

This is illustrated in Figure 7.2 where, for $k = 1$, the similarity between the target case T_1 and its NLN is much higher than the similarity between T_1 and its NUN, whereas the similarity between target case T_2 and its NLN is only marginally higher than the similarity between T_2 and its NUN. The ratio of these similarites for T_1 will give a higher result than that for T_2 indicating higher confidence in the classification of T_1 than T_2.

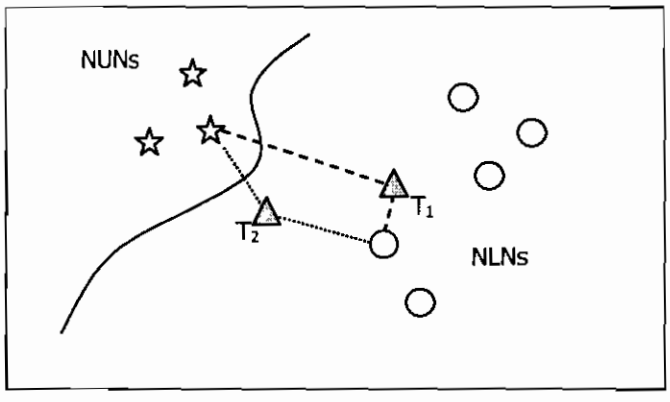


Figure 7.2: Similarity Ratio Confidence Measure

Similarity Ratio Within K

The Similarity Ratio Within K is similar to the Similarity Ratio as described above except that, rather than consider the first k NLNs and the first k NUNs of a target case t, it only uses the NLNs and NUNs from the first k neighbours. It is defined in Equation 7.3.

$$SimRatio(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NN_i(t)) \,\mathrm{One}(t, NN_i(t))}{1 + \sum_{i=1}^{k} Sim(t, NN_i(t))(1 - \mathrm{One}(t, NN_i(t)))} \tag{7.3}$$

where $Sim(a,b)$ is the calculated similarity between cases a and b and $\mathrm{One}(a,b)$ returns one if the class of a is the same as the class of b or zero otherwise.

This measure will attempt to reward cases that have no NUNs within the first k neighbours, i.e. are in a cluster of k cases of the same class. This is illustrated in Figure

7.3 where, considering $k = 3$, the target case T_1 has no NUNs within the first three neighbours whereas target case T_2 has two NUNs and one NLN. The Similarity Ratio Within K will be much larger for T_1 than that for T_2 indicating higher confidence in the classification of T_1 than T_2.

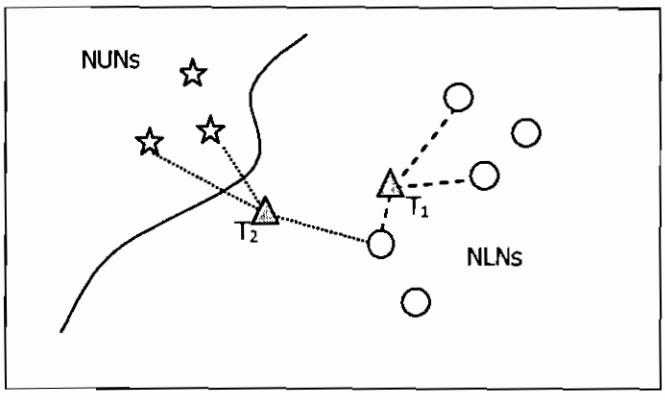


Figure 7.3: Similarity Ratio Within K Confidence Measure

If a target case t has no NUNs then Equation 7.3 is effectively Equation 7.2 with the denominator set to one.

Sum of NN Similarities

The Sum of NN Similarities measure is the total similarity of the NLNs in the first k neighbours of the target case t, see Equation 7.4.

$$SumNNSim(t,k) = \sum_{i=1}^{k} \text{One}(t, NN_i(t)) Sim(t, NN_i(t)) \tag{7.4}$$

where $Sim(a,b)$ is the calculated similarity between cases a and b and $\text{One}(a,b)$ returns one if the class of a is the same as the class of b or zero otherwise.

For target cases in a cluster of cases of similar class this number will be large. For cases which are closer to the decision surface and have NUNs within the first k neighbours, this measure will be smaller. In fact for target cases with no NUNs within the first k neighbours this measure will be equal to the value of the Similarity Ratio Within K. Although this measure does not reward such cases as strongly as the Similarity Ratio Within K does as the resulting measure for the sum of the NLNs is not reduced by the influence of the NUNs.

Average NN Similarity

The Average NN Similarity measure is the average similarity of the NLNs in the first k neighbours of the target case t, see Equation 7.5.

$$SumNNSim(t,k) = \frac{\sum_{i=1}^{k} \text{One}(t, NN_i(t)) Sim(t, NN_i(t))}{\sum_{i=1}^{k} \text{One}(t, NN_i(t))} \tag{7.5}$$

where $Sim(a,b)$ is the calculated similarity between cases a and b and $\text{One}(a,b)$ returns one if the class of a is the same as the class of b or zero otherwise.

7.1.2 Assessing k-NN Confidence Measure Performance

In order to assess the performance of these confidence measures we evaluated each of them on datasets 1 to 5 as described in Section 5.1.1. Case-bases were built from each of these datasets and then edited using the CBE case-base editing procedure. After editing the datasets averaged 700 emails in size with an average of 45% spam and 55% legitimate emails.

The evaluation involved performing a leave-one-out validation on each dataset for each measure. We evaluated each measure using k neighbours from $k = 1$ up to $k = 15$ and identified the confidence threshold, over all the k values, that gave us the highest proportion of correctly predicted spam emails when there were no incorrect predictions (i.e. FPs). This is illustrated in Figure 7.4.

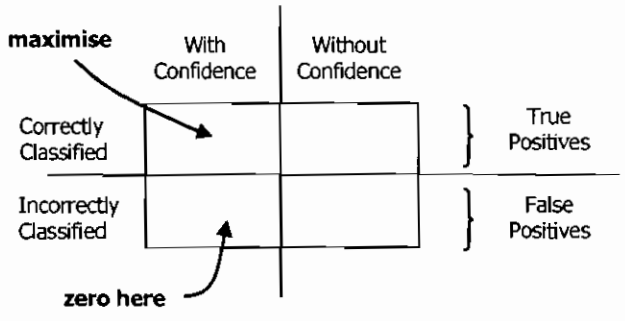


Figure 7.4: Criteria used to identify the best confidence threshold level

This was achieved by recording the confidence measure results for each target case $c_i, i = 1 \ldots N$, that was classified by ECUE as spam. The results recorded included the number of neighbours k used in the measure, whether the target case was classified correctly or not and the measure calculated, m_{ik}. Setting the threshold t_k equal to the minimum value of m_{ik} for a given k and varying the threshold in small units ($t_k = t_k +$

.01) up to the maximum value of m_{ik}, the number classified correctly with confidence (CC_k) and the number classified incorrectly with confidence (CI_k) were calculated, where confidence exists for case c_i when $m_{ik} > t_k$. The selected threshold value was the threshold t_k that maximised CC_k, the number of spam correctly predicted with high confidence when the number of incorrect predictions with high confidence was zero (i.e. $CI_k = 0$). This is effectively identifying the specific value t for the confidence measure such that all spam correctly classified have a higher value than t and those spam which are classified incorrectly have a lower value than t, as illustrated in Figure 7.5.

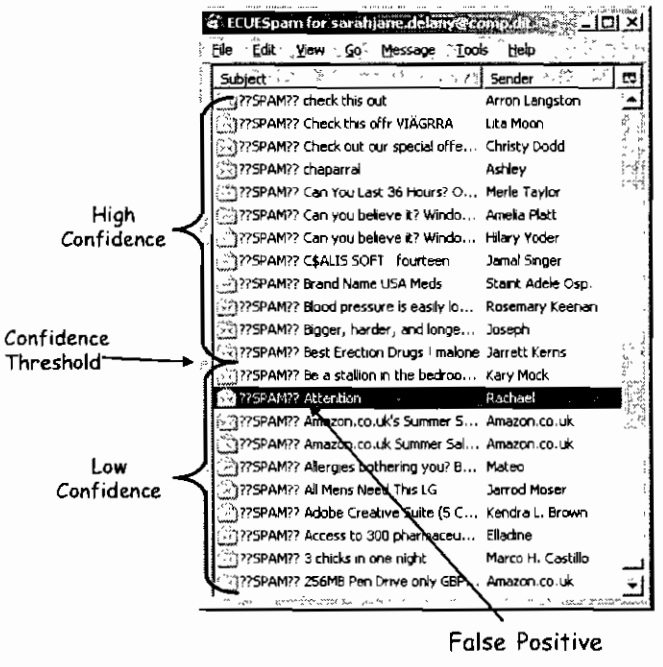


Figure 7.5: Illustration of the confidence threshold level on a spam mail folder.

The results of this evaluation are presented in rows 1 to 5 of Table 7.1 (the other measures in rows 6 to 9 are described later). It details for each measure the highest percentage confidence that can be achieved on each dataset. This is the proportion of *spam* predictions that are made with high confidence. In all situations no highly confident incorrect predictions were made so no FPs are included in this proportion. In effect, this proportion of the spam can be ignored by the user, whereas the remaining percentage would have to be checked by the user.

Looking at the proportion of spam predictions for which confidence is high across all datasets it is evident that no single measure achieves good percentage confidence across all datasets. If we define "good" performance as having confidence in at least 50% of the spam predictions, none of the measures achieve "good" performance on more than three

Table 7.1: Best percentage confidence achievable for each dataset using different confidence measures

Confidence Measure	**Dataset 1**	**Dataset 2**	**Dataset 3**	**Dataset 4**	**Dataset 5**	**Avg**
Avg NUN Index	23%	76%	75%	41%	44%	51.8%
Sim Ratio	46%	84%	50%	49%	16%	49.0%
Sim Ratio Within k	21%	29%	71%	91%	57%	54.8%
Sum NN Sim	21%	29%	68%	91%	58%	53.4%
Avg NN Sim	20%	29%	49%	91%	60%	49.8%
Naive Bayes	0%	94%	0%	83%	56%	46.4%
SVM	29%	100%	77%	81%	33%	63.8%
Logistic Regression	0%	97%	22%	18%	0%	27.4%
ACM	55.4%	85.4%	83.8%	93.7%	77.3%	79.1%

of the five datasets. The best performing measure is the Similarity Ratio Within K which has good performance on three of the five datasets with an average performance across all datasets of 54.8% but with minimum performance of 21%.

7.1.3 Naïve Bayes, SVM and Logistic Regression Confidence Measures

Naïve Bayes, Logistic Regression and SVM classifers produce numeric scores; Naïve Bayes and Logistic Regression produce a 'probability' of spam whereas an SVM produces a 'distance' from the byperplane separating the spam and non spam classes. These scores can be used to predict confidence in the classifiers' prediction.

We examined confidence measures produced by Naïve Bayes, Logistic Regression and an SVM on the five datasets. The Naïve Bayes implementation used was that described in Section 3.4.1. The SVM implementation used was a 2-norm soft-margin SVM with a normalised dot product kernel function as described in Section 3.4.2. The Logistic Regression implementation used was that described in Nugent et al. (2005).

In all cases, the confidence threshold was identified as the highest nnmeric score returned by the classifier for an FP prediction. This ensured that no incorrectly classified spam emails were considered confident predictions. The sixth row of Table 7.1 gives the confidence predictions for the five datasets using the Naïve Bayes classifier. It is clear from the results that the Naïve Bayes numeric score cannot be used as a predictor of confidence. In two of the five datasets there are zero confident predictions as there are FPs with the maximum score.

The seventh row of the Table 7.1 gives the confidence predictions for the five datasets using an SVM for classification. Although the average score across all datasets of 63.8% is

higher than the best of the k-NN measures the SVM confidence measure does not realistically achieve any better overall performance as it also only achieves "good" performance on three of the five datasets but with slightly higher minimum performance of 29%. It is worth noting that the performance of dataset 2 is actually 99.7% but is reported as 100% due to rounding.

The eighth row of the Table 7.1 gives the confidence predictions for the five datasets using Logistic Regression for classification. The results show that this classifier returned the lowest overall average confideuce and included zero confideut predictions for two of the five datasets.

7.1.4 Implications for Predicting Confidence in Spam Filtering

To summarise, it appears that the confidence measures for k-NN, Naïve Bayes, SVMs and Logistic Regression presented here cannot consistently produce estimates of prediction confidence for spam. The average performance of the k-NN and the SVM measures shows promise however the lack of consistency across all datasets is an issue. The thresholds achieved for each k-NN ueasure across the five datasets also varies considerably. For exaınple, considering the Similarity Ratio Within K measure, which has the best of the k-NN measures performance, Table 7.2 shows the variation in the threshold across the five datasets.

Table 7.2: Demonstrating the variation in thresholds for the Similarity Within K Ratio confidence measure across the five datasets

Threshold	**Dataset 1**	**Dataset 2**	**Dataset 3**	**Dataset 4**	**Dataset 5**
k (num neighbours used)	11	7	14	1	3
Value	991.07	574.08	717.04	58	214.1

It is important to note that the figures in Table 7.1 are very optimistic as the test data was used to set the threshold.

7.2 The Aggregated Confidence Measure

Since none of the individual measures discussed in Section 7.1.2 was consistently effective at predicting confidence we evaluated a number of aggregation approaches which involved combining the results from the individual measures. The aggregation approaches we considered included:

(i) Summing the results from each of the 5 individual measures evaluated at the same value of k and comparing the sum against a threshold;

(ii) Using the best threshold for each individual measure and indicating confidence if a certain number of the measures indicate confidence;

(iii) Using a fixed k across all measures and indicating confidence if a certain number of the measures indicate confidence.

We found that the simplest and most effective method of aggregating the results is to assign confidence to a prediction if any of the individual measures indicated that the prediction was confident as in (ii) above. We call this measure the Aggregated Confidence Measure (ACM). The algorithm for the ACM has two stages:

(i) calculation of the constituent measure threshold values in a pre-classification stage,

(ii) determination of the ACM during classification.

The pre-classification stage involves pre-processing of the case-base to identify the best threshold for each individual constituent measure. This is performed in the manner described in Section 7.1.2. A threshold consists of two values; the k value indicating the number of neighbours to use in the calculation and the actual threshold value above which the prediction is considered confident. These constituent measure thresholds are stored.

The ACM is then determined during classification for each target case that is classified as spam by ECUE. Using the appropriate threshold value of k, the actual score for each individual constituent measure is calcuated for the target case. The ACM specifies that if at least one of the calculated scores for the individual measures is equal to or greater than the stored threshold value for that measure, confidence is expressed in the prediction.

7.2.1 Assessment of ACM's Performance

We evaluated the ACM on the five datasets already used in Section 7.1.2. The results are presented in row 9 of Table 7.1. Unlike its constituent measures, the ACM is effective across all datasets with an average of 79% of the spam predictions being predicted with high confidence. The ACM also results in more than 50% of each dataset being predicted with high confidence. It is worth noting that the level of high confidence predictions for the ACM is also higher than the best individual measure's performance on each dataset (rows 1 to 5 of Table 7.1).

7.2.2 Evaluation on Unseen Data

One limitation of the evaluation performed in Section 7.2.1 is that the datasets on which the assessment was performed were themselves used to derive the confidence thresholds for the constituent confidence measures. In order to validate the ACM it is necessary to evaluate its performance on unseen data. To do this we used datasets 6 and 7 (see Section 5.1.1). These datasets each include a training set of 1000 emails and also eight and six months of test emails repectively. The monthly class distribution of the test emails is evident in rows 2 and 3 of Tables 7.3 and 7.4.

To evaluate the ACM on unseen data involved building confidence thresholds for the ACM constituent measures on the initial case-base and then classifying the remaining emails using the ACM to determine how confident the *spam* predictions are. In this way, the test emails were not used in the determination of the confidence thresholds in any way.

The test emails were presented in date order for classification. Since this email data is subject to concept drift, ECUE's case-base update policy was applied to allow the classifier to learn from the new types of spam and legitimate email presented. In order to keep the confidence thresholds in line with the updates to the case-base an update policy for the confidence thresholds was also applied. The confidence threshold update policy had two components; firstly the confidence thresholds were updated whenever a FP email occurred as the initial confidence threshold levels may not be appropriate once more FPs have been added to the training set and secondly after a monthly feature reselect.

The existing ECUE update policy discussed in Section 5.5 was extended for this evaluation. In addition to the two existing components; the daily update of the case-base with any misclassifieds emails that occurred that day and the monthly feature reselection process to allow the case representation to take any new predictive features into account, a new update component was added. This involved an immediate update of the case-base with all misclassified emails when a FP occurred and was necessary to keep the update policies in line with each other.

Tables 7.3 and 7.4 show the results of testing the performance of the ACM on unseen data using the two datasets 6 and 7. The tables present the accumulated monthly results for each dataset listing the total number and types of emails that were classified, the percentage of incorrect spam predictions (i.e. FPs) made (labelled *%FP classified*) and the percentage of incorrect spam predictions made with high confidence (labelled *%Confident FPs*). The table also gives the total percentage of spam predictions with high confidence

(labelled *%Confidence*).

Table 7.3: Performance of ACM on unseen data using dataset 6

Month	1	2	3	4	5	6	7	8	Overall
Total emails classified	772	542	318	1014	967	1136	1370	1313	7382
Number of Spam	629	314	216	925	917	1065	1225	1205	6496
Number of Non Spam	93	228	102	89	50	71	145	108	886
%FPs classified	4.3%	2.6%	1.0%	1.1%	6.0%	1.4%	0.0%	1.9%	2.0%
%Confident FPs	0.0%	0.9%	0.0%	1.1%	0.0%	0.0%	0.0%	0.9%	0.5%
%Confidence	70%	87%	76%	94%	89%	73%	77%	99%	85%

Table 7.4: Performance of ACM on unseen data using dataset 7

Month	1	2	3	4	5	6	Overall
Total emails classified	293	447	549	693	534	495	3011
Number of Spam	142	391	405	459	406	476	2279
Number of Non Spam	151	56	144	234	128	19	732
%FPs classified	0.7%	3.6%	3.5%	2.6%	1.6%	0.0%	2.2%
%Confident FPs	0.0%	3.6%	0.7%	0.4%	1.6%	0.0%	0.8%
%Confidence	95%	95%	87%	64%	89%	88%	85%

In both datasets predictions of confidence are high, averaging 85% in both cases with a lowest monthly level of 64%. This is the percentage of spam predictions that can be ignored by the user, the remaining spam predictions can either be flagged in the Inbox as *Maybe Spam* or placed in a separate *Maybe Spam* folder for the user to check.

However in some of the months the ACM has resulted in confident incorrect predictions. Although the actual numbers of emails are low (four emails for Dataset 6 and six emails for Dataset 7) the ideal situation is one where all incorrect predictions have low confidence aud will be flagged for the user to check. FPs flagged as confident will end up in the *spam* folder and may be missed by the user. Examining the confident FPs, three are emails from mailing lists and two are responses to Web registrations which users may not be too concerned with missing. The remaining five are important, some work related and one even a quotation in response to a online car hire request.

It is clear that we are approaching the limits of the accuracy of machine learning techniques in this domain. We see two possibilities for addressing these FPs. Close examination of such emails may identify domain specific characteristics that could be used as a feature or number of features in the case representation. Secondly, most deployed spam

filtering solutions do not rely on one approach for filtering spam, they combine a number of techniques including white and black listing, rules, collaborative and learning approaches. Incorporating additional techniques into ECUE to add to its case-based approach could help in catching these outlier FPs.

7.3 Conclusions

In this chapter we have shown that confidence measures based on the numeric scores from Naïve Bayes, SVM, Logistial Regression or measures based on the k nearest neighbours for a case-based classifier are not consistent at predicting confidence in the spam filtering domain.

We have described an aggregation-based approach to combining individual k-NN confidence measures that shows great promise in confidently predicting spam. We evaluated this aggregated confidence measure by incorporating it into the classification process of a case-based spam filter and showed that it could successfully separate the spam predictions into two sets, those with high confidence of spam which can be ignored by the user and those with low confidence which should be periodically checked for False Positives (Delany et al. 2005c). The high-confidence set included 85% of the predicted spam reducing the number of spam that the user needs to check.

Chapter 8

CONCLUSIONS and FUTURE WORK

In spite of considerable efforts to reduce and stop spam, email users are still faced with significant numbers of unsolicited and unwanted email arriving in their Inbox. The difficulties in identifying spam email arise mainly from the fact that spam is constantly changing; spammers try to make it resemble legitimate email to allow it to bypass the many filters used to trap it. The challenge of tracking this concept drift makes spam filtering an incremental learning problem. We have tackled this problem in this thesis and the contributions made fall under two headings, the contributions to spam filtering and the contributions to Machine Learning.

Contributions to Spam Filtering

The main contribution under this heading is the development of ECUE, a case-based spam filtering application that learns from new examples of spam and legitimate email. As a lazy, local learner CBR offers distinct advantages over alternative eager approaches to spam filtering such as Naïve Bayes or Support Vector Machines, approaches that are more common in commercial filters. It provides capabilities to learn seamlessly without the need for a separate learning process. Also, the fact that spam is a diverse concept makes CBR, a local learner, an appropriate choice.

We evalnated ECUE in two main ways; offline evaluations on a number of pre-compiled email datasets and online evaluations where we assessed a real-time online system that worked along-side IMAP mail readers to filter the mail received at an individual's personal mailbox. Both types of evaluation demonstrate ECUE's effectiveness at filtering spam email. The online evaluations showed that ECUE was successful at filtering mail at a personal level, identifying between 92% and 94% of spam correctly with less than 1.0% false positives identified. For users who used ECUE as a second-level spam defence, operating

the filter on email that had already been passed through a organisation-level gateway spam filter, ECUE still successfully filtered between 48% and 66% of the spam correctly albeit with a slightly higher false positive rate for a small number of users of just over 1%.

We contend that machine learning will be key to fighting spam, a view supported by others[1]. As techniques which utilise a probabilistic classifier, such as Naïve Bayes, to detect spam email are already patented[2], it is necessary to find other techniques which offer at least comparable results. Our evaluations discussed in this thesis show that using a k-NN classifier for spam filtering is certainly no worse than using Naïve Bayes. In fact, our research suggests that CBR demonstrates better performance for learning over time.

A spam filtering system such as ECUE needs user intervention to indicate when mistakes have been made to allow the update policy to be triggered. This places quite an onus on the user of the system to check the predictions made by the system for mistakes. A further contribution to spam filtering is the development of a confidence measure that reduces this effort. Those spam predictions made with high confidence could be ignored by the user and only those predictions with lower confidence need to be checked for correct classification. We have described an aggregation-based approach to combining individual k-NN confidence measures that shows great promise in confidently predicting spam. We evaluated this aggregated confidence measure by incorporating it into the classification process of a case-based spam filter and showed that it could successfully separate the spam predictions into two sets, those with high confidence of spam which can be ignored by the user and those with low confidence which should be periodically checked for False Positives. The high-confidence set included 85% of the predicted spam, reducing the number of spam that the user needs to check.

The consequences of False Positives make spam filtering a challenging area for Machine Learning. There are other document and message classification areas where False Positives do not have the same significance as in spam filtering. These include message filtering and routing in such application areas as mobile short message services (SMS), customer support and help desks. The general ECUE framework could be very effective at developing solutions in these areas. The technology could also be applied to identification of inappropriate conteut in messages and monitoring of documents or messages for regulatory compliance.

[1] http://www.research.microsoft.com/ joshuago/spamconferenceshort.ppt
[2] United States Patent 6,161,130 2000

Contributions to Machine Learning

Our objective in this thesis was the application of instance-based learning to handle concept drift. We chose the domain of spam filtering and demonstrated that spam filtering is a classification problem with significant concept drift. Our most significant contribution to Machine Learning in this thesis is the development of a case-base maintenance strategy that can handle concept drift in spam filtering. The two components of this strategy include:

- a novel case-base editing algorithm to remove noisy and exceptional cases and
- a case-base update policy to allow the addition of new training examples to the case-base as they are encountered.

A key component in any machine learning based spam filter is a procedure to manage the large amounts of training data (namely the volumes of legitimate and spam email received by individuals). We presented in this thesis a new case-editing technique, Competence Based Editing (CBE). CBE is itself a contribution to Machine Learning. It has two stages, a noise reduction phase called Blame Based Noise Reduction and a redundancy elimination phase called Conservative Redundancy Reduction.

Our noise reduction algorithm, BBNR, focuses on the damage that certain cases are causing in classifications. Traditional noise reduction mechanisms tend to focus on removing the actual cases that are misclassified. In contrast to traditional approaches, we attempt to identify those cases causing the misclassifications and use this information coupled with how useful the case is in classification to identify training cases we would be better off without. Comparative evaluations of this algorithm with the traditional noise reduction techniques have shown an improved performance across all datasets used in the evaluation. Experiments incorporating BBNR into existing competence based editing techniques have shown that BBNR improves the performance of all these techniques over the datasets on which it was evaluated.

Past research into case-base editing was motivated by the need for speed and the objective was often to maintain competence while reducing the size of the case-base. Now, with speed less of an issue due to the advances in hardware and system software technologies, we find an opportunity to improve competence if we relax existing constraints. Our redundancy reduction process (CRR) was motivated by the observation that state-of-the-art techniques were inclined to be too aggressive in removing cases and tended to result

in some loss of generalisation accuracy, at least in the domain of spam filtering. This is in effect a tendency to overfit the training data by finding minimal sets that cover the data. CRR is much more conservative in removing cases and produces larger edited case-bases that have the best generalisation accuracy in this domain.

Handling concept drift is an area that warrants, and has seen, considerable research. We have contended that the approaches to tracking concept drift can be categorised as instance selection, instance weighting and ensemble learning. The most effective approaches include instance selection and ensemble learning methods. ECUE's approach for handling concept drift falls under the category of instance selection although it doesn't follow the most common instance selection technique applied, that of windowing. In this thesis we compared our instance selection approach with that of windowing and also with ensemble learning. These evaluations show that the case-management (instance selection) approach to handling concept drift performs better than the typical window-based approach and is more straightforward and as effective as the ensemble alternatives that we considered. We claim that the discriminating power of the single classifier solution is better than that of the ensemble techniques with the most effective ensemble technique being the one where the best ensemble members are selected based on an assessment of their performance on recent data. Some of the ensemble techniques return very strong results on FPs, but this comes at a significant cost in overall accuracy. We have pointed out that this reflects the greater potential there is to control the bias of the ensemble classifier away from FPs.

We have emphasised the importance of being able to attach confidence values to predictions in CBR. Surprisingly we have shown that classification scores from ranking classifiers such as Support Vector Machines, Nearest Neighbour Classifiers, Naive Bayes and Logistic Regression are poor estimates of classification confidence. A contribution of this thesis to Machine Learning is an alternate aggregation-based approach to combining individual k-NN confidence measures that shows great promise in confidently predicting spam.

To conclude, there is no single approach that will be 100% effective at handling spam. The solution to spam is currently a multi-layered approach, utilising legislative measures, authentication techniques and filtering. Filtering plays and will continue to play a significant role in this fight against spam. We believe that ECUE, our case-based approach to filtering can be a important contributor to content-based spam filtering. We have shown how it can handle the changes in spam emails with relative ease without putting too much burden on the individual using it.

8.1 Future Work

In this thesis we presented a noise reduction technique (BBNR) that was more effective than the more traditional Wilson based noise-reduction techniques at identifying and removing noise in datasets. Our preliminary evaluations of BBNR on other types of datasets, outside of the spam filtering domain, have indicated that it is not as effective in these areas. We plan to continue work along this line to identify what characteristics of email allow BBRN to be effective in certain domains and not as effective in others.

During our evaluations the SVM demonstrated strong classification power for the static datasets. We did not proceed with this line of research in this thesis due to the difficulty in updating a SVM for each new batch of data. We will pursue research on updatable SVMs (Syed et al. 1999, Klinkenberg and Joachims 2000, Rüping 2001) in future work comparing their performance with that of the case-based approach.

In this thesis we compared a variety of ensemble techniques with our ECUE instance selection technique for handling concept drift. Before abandoning the use of ensembles on this problem we propose to consider a more complex integration strategy. For instance, a variant of dynamic integration as described by Tsymbal and Puuronen (2000) can be used. In addition we propose to evaluate weighted ensemble members such as those used by Kolter and Maloof (2003) and Stanley (2003) and build ensemble members that cover different parts of the problem space (most likely using a clustering algorithm) rather than depending on members that cover particular time periods.

In addition to the further work in the area of spam filtering just discussed, the technology presented in this thesis has applicability in other areas related to document and message classification which were already mentioned above. Further work will include evaluating the technology in these other application areas.

Bibliography

Aamodt, A. and Plaza, E.: 1994, Case-based reasoning: foundational issues, methodological variations and system approaches, *AI Communications* **7**(1), 39–59.

Aha, D. W.: 1997, Editorial, *Artificial Intelligence Review, Special Issue on Lazy Learning* **11**(1-5), 7–10.

Aha, D. W., Kibler, D. and Albert, M. K.: 1991, Instance-based learning algorithms, *Machine Learning* **6**, 37–66.

Ahmed, S. and Mithun, F.: 2004, Word stemming to enhance spam filtering, *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS'04)*.

Albrecht, K., Burri, N. and Wattenhofer, R.: 2005, Spamato - an extendable spam filter system, *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS'05)*.

Alpaydin, E.: 2004, *Introduction to Machine Learning*, The MIT Press.

Androutsopoulos, I., Koutsias, J., Chandrinos, G., Paliouras, G. and Spyropoulos, C.: 2000a, An evaluation of naive bayesian anti-spam filtering, *in* G. Potamias, V. Moustakis and M. van Someren (eds), *Procs of Workshop on Machine Learning in the New Information Age, ECML 2000*, pp. 9–17.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. and Spyropoulos, C.: 2000b, An experimental comparison of naive bayesian and keyword-based anti-spam filtering with encrypted personal email messages, *in* N. Belkin, P. Ingwersen and M. Leong (eds), *Procs of 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 160–167.

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C. and Stamatopoulos, P.: 2000c, Learning to filter spam email: A comparison of a naive bayesian

and a memory based approach, *in* H. Zaragoza, P. Gallinari and M. Rajman (eds), *Proceedings of the Workshop on Machine Learning and Textual Information Access, PKDD 2000*, pp. 1–13.

Androutsopoulos, I., Paliouras, G. and Michelakis, E.: 2000d, Learning to filter unsolicited commercial email, *Technical Report 2004/02*, NCSR Demokritos.
URL: *http://www.iit.demokritos.gr/skel/i-config/publications*

Ashley, K. D. and Aleven, V.: 1991, Toward an intelligent tutoring system for teaching law students to argue with cases, *ICAIL '91: Proceedings of the 3rd international conference on Artificial intelligence and law*, ACM Press, New York, NY, USA, pp. 42–52.

Atkeson, C. G., Moore, A. W. and Schaal, S.: 1997, Locally weighted learning, *Artificial Intelligence Review* **11**(1-5), 11–73.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A.: 1999, *Modern Information Retrieval*, ACM Press / Addison-Wesley.

Bellman, R.: 1961, *Adaptive Control Processes*, Princeton University Press.

Berger, A., Pietra, S. D. and Pietra, V. D.: 1996, A maximum entropy approach to natural language processing, *Computational Linguistics* **22**(1), 39–71.

Blair, D. C. and Maron, M. E.: 1990, Full-text information retrieval: further analysis and clarification, *Information Processing and Management* **26**(3), 437–447.

Bottou, L. and Vapnik, V.: 1992, Local learning algorithms, *Neural Computation* 4(6), 888–900.

Boykin, P. and Roychowdhury, V.: 2005, Personal email networks: An effective anti-spam tool, *IEEE Computer* **38**(4), 61–68.

Bradley, A.: 1997, The use of the area under the ROC curve in the evaluation of machine learning algorithms, *Pattern Recognition* **30**, 1145–1150.

Breiman, L.: 1996, Bagging predictors, *Machine Learning* **24**(2), 123–140.

Breiman, L.: 1999, Pasting small votes for classification in large databases and online, *Machine Learning* **36**(1-2), 85–103.

Brighton, H. and Mellish, C.: 2002, Advances in instance selection for instance-based learning algorithms., *Data Mining and Knowledge Discovery* **6**(2), 153–172.

Brodley, C.: 1993, Addressing the selective superiority problem: Automatic algorithm/mode class selection, *Proceedings of the 10th International Conference on Machine Learning (ICML 93)*, Morgan Kaufmann Publishers Inc., pp. 17–24.

Bruninghaus, S. and Ashley, K.: 1999, Bootstraping case-based development with annotated case summaries, *Case-based reasoning research and applications*, number 1650 in *LNCS*, Springer Verlag, pp. 59–73.

Bruninghaus, S. and Ashley, K.: 2001, The role of information extraction for textual cbr, *Case-based reasoning research and development*, number 2080 in *LNCS*, Springer Verlag, pp. 74–89.

Bruninghaus, S. and Ashley, K.: 2003, Combining model-based and case-based reasoning for predicting the outcomes of legal cases, *Case-based reasoning research and development*, number 2689 in *LNCS*, Springer Verlag, pp. 65–79.

Cameron-Jones, R. M.: 1992, Minimum description length instance-based learning, *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., pp. 368–373.

Cardoso-Cachopo, A. and Oliveira, A.: 2003, An empirical comparison of text categorisation methods, *in* M. A. Nascimento, E. S. de Moura and A. L. Oliveira (eds), *Proceedings of Conference on String Processing and Information Retrieval*, Springer Verlag, pp. 183–196.

Carreras, X. and Márquez, L.: 2001, Boosting trees for anti-spam email filtering, *Proceedings of RANLP-01, 4th International Conferenee on Recent Advances in Natural Language Processing*, Tzigov Chark, BG.

Ceglowski, M., Coburn, A. and Cuadrado, J.: 2003, Semantic search of unstructured data using contextnal network graphs.

Cheetham, W.: 2000, Case-based reasoning with confidence., *in* E. Blanzieri and L. Portinale (eds), *5th European Workshop on Case-Based Reasoning*, Vol. 1898 of *LNCS*, Springer, pp. 15–25.

Cheetham, W. and Price, J.: 2004, Measures of solution accuracy in case-based reasoning systems, *in* P. Funk and P. González-Calero (eds), *7th European Conference on Case-Based Reasoning (ECCBR 2004)*, Vol. 3155 of *LNAI*, Springer, pp. 106–118.

Christianini, N. and Shawe-Taylor, J.: 2000, *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*, Cambridge University Press.

Chuan, Z., Xianliang, L., Mengshu, H. and Xu, Z.: 2005, An lvq-based neural network anti-spam email approach, *SIGOPS Operating Systems Review* **39**(1), 34–39.

Cohen, W. W.: 1995, Fast effective rule induction, *in* A. Prieditis and S. Russell (eds), *Proc. of the 12th International Conference on Machine Learning*, Morgan Kaufmann, pp. 115–123.

Coyle, L., Doyle, D. and Cunningham, P.: 2004, Representing similarity for CBR in XML, *in* P. Funk and P. A. González-Calero (eds), *Advances in Case-Based Reasoning, 7th European Conference, (ECCBR 2004)*, Vol. 3155 of *Lecture Notes in Computer Science*, Springer, pp. 119–127.

Cristiani, N. and Scholkopf, B.: 2002, Support vector machines and kernel methods: The new generation of learning machines, *AI Magazine* **23**(3), 31–41.

Cunningham, P. and Carney, J.: 2000, Diversity versus quality in classification ensembles based on feature selection., *in* R. L. de Mántaras and E. Plaza (eds), *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Proceedings*, Vol. 1810 of *LNCS*, Springer, pp. 109–116.

Cunningham, P., Finn, D. and Slattery, S.: 1994, Knowledge engineering requirements in derivational analogy, *Topics in Case-based Reasoning* **837**, 234–245.

Cunningham, P., Nowlan, N., Delany, S. and Haahr, M.: 2003, A case-based approach to spam filtering that can track concept drift, *ICCBR 2003 Workshop on Long-Lived CBR Systems*.

Damiani, E., di Capitani di Vimercati, S., Paraboschi, S. and Samarati, P.: 2004, P2p-based collaborative spam detection and filtering, *4th International Conference on Peer-to-Peer Computing (P2P'04)*, pp. 176–183.

Davis, R.: 1982, Expert systems: Where are we? and where do we go from here?, *AI Magazine* **3**(2), 3–22.

Davis, R. and Buchanan, B.: 1985, Meta level knowledge, *in* F. Hayes-Roth, D. A. Waterman and D. B. Lenat (eds), *Rule-Based Expert Systems*, Addison-Wesley, London, pp. 507–530.

Delany, S. J. and Cunningham, P.: 2004, An analysis of case-based editing in a spam filtering system, *in* P. Funk and P. González-Calero (eds), *7th European Conference on Case-Based Reasoning (ECCBR 2004)*, Vol. 3155 of *LNAI*, Springer, pp. 128–141.

Delany, S. J., Cunningham, P. and Coyle, L.: 2004a, An assessment of case-based reasoning for spam filtering, *in* L. McGinty and B. Crean (eds), *Procs. of 15th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 9–18.

Delany, S. J., Cunningham, P. and Coyle, L.: 2005a, An assessment of case-based reasoning for spam filtering, *Artificial Intelligence Review* **24**(3–4), 359–378.

Delany, S. J., Cunningham, P. and Doyle, D.: 2005c, Generating estimates of classification confidence for a case-based spam filter, *Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005)*, Vol. 3620 of *LNAI*, Springer, pp. 170–190.

Delany, S. J., Cunningham, P., Tsymbal, A. and Coyle, L.: 2004b, A case-based technique for tracking concept drift in spam filtering, *in* A. Macintosh, R. Ellis and T. Allen (eds), *Applications and Innovations in Intelligent Systems XII, Procs. of AI 2004*, Springer, pp. 3–16.

Delany, S. J., Cunningham, P., Tsymbal, A. and Coyle, L.: 2005b, A case-based technique for tracking concept drift in spam filtering, *Knowledge-Based Systems* **18**(4–5), 187–195.

Dietterich, D. T.: 1998, Approximate statistical tests for comparing supervised classification learning algorithms, *Neural Computing* **10**, 1895–1923.

Doyle, D., Loughrey, J., Nugent, C., Coyle, L. and Cunningham, P.: 2005, Fionn: A framework for developing CBR systems, *Expert Update* **8**(1), 11–14.

Drucker, H., Wu, D. and Vapnik, V.: 1999, Support vector machines for spam categorisation, *IEEE Transactions on Neural Networks* **10**(5), 1048–1055.

Dumais, S., Platt, J., Heckerman, D. and Sahami, M.: 1998, Inductive learning algorithms and representations for text categorisation, *Procs of ACM 7th Internation Conference on information and Knowledge Management (CIKM 98)*, ACM Press, pp. 148–155.

Fausett, L.: 1993, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice Hall.

Fawcett, T.: 2003, "in vivo" spam filtering: a challenge problem for kdd, *SIGKDD Explorations* **5**(2), 140–148.

Ferreri, J.: 1999, *Entrepreneur Magazine's Knock-Out Marketing:Powerful strategies to punch up your sales*, Entrepreneur Press.

Foltz, P., Laham, D. and Landauer, T.: 1999, Automatic essay scoring: application to educational technology, *in* B. Collins and R. Oliver (eds), *Proceedings of EdMedia*.

Frakes, W. B. and Baeza-Yates, R. A. (eds): 1992, *Information Retrieval: Data Structures & Algorithms*, Prentice-Hall.

Frank, E. and Witten, I. H.: 1998, Generating accurate rule sets without global optimization, *Proc. 15th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 144–151.

Freund, Y. and Schapire, R.: 1999, A short introduction to boosting, *Journal Japanese Society for Artificial Intelligence* **14**(5), 771–780.

Gates, G. W.: 1972, The reduced nearest neighbor rule, *IEEE Transactions on Information Theory* **18**(3), 431–433.

Gee, K. R.: 2003, Using latent semantic indexing to filter spam, *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, ACM Press, pp. 460–464.

Gray, A. and Haahr, M.: 2004, Personalised, collaborative spam filtering, *Proceedings of 1st Conference on Email and Anti-Spam*.

Gupta, K. M. and Aha, D. W.: 2004, Towards acquiring case indexing taxonomies from text., *in* V. Barr and Z. Markov (eds), *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, AAAI Press.

Hart, P. E.: 1968, The condensed nearest neighbor rule, *IEEE Transactions on Information Theory* **14**(3), 515–516.

Hidalgo, J. M. G.: 2002, Evaluating cost-sensitive unsolicited bulk email categorization, *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, ACM Press, New York, NY, USA, pp. 615–620.

Hidalgo, J. M. G., López, M. M. and Sanz, E.: 2000, Combining text and heuristics for case-sensitive spam filtering, *Proceedings of the 4th Computational Natural Language Learning Workshop (CONLL-2000)*.

Hosmer, D. W. and Lemeshow, S.: 2000, *Applied Logistic Regression*, Wiley Series in Probability and Statistics, Wiley.

Hovold, J.: 2005, Naïve bayes spam filtering using work-position-based attributes, *Proceedings of the 2nd Conference on Email and Anti-Spam*.

Hughes, L.: 1998, *Internet e-mail: protocols, standards and implementation*, Artech House Inc.

Joachims, T.: 1998, Text categorization with support vector machines: learning with many relevant features, *in* C. Nédellec and C. Rouveirol (eds), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in *LNCS*, Springer Verlag, Heidelberg, DE, pp. 137–142.

Judge, P., Alperovitch, D. and Yang, W.: 2005, Understanding and reversing the profit model of spam, *WEIS 05, Workshop on the Economics of Information Security*.

Kilgarriff, A. and Salkie, R.: 1996, Corpus similarity and homogeneity via word frequency, *Proceedings of EURALEX '96*.

Klinkenberg, R.: 2004, Learning drifting concepts: Example selection vs. example weighting, *Intelligent Data Analysis* **8**(3).

Klinkenberg, R. and Joachims, T.: 2000, Detecting concept drift with support vector machines, *in* P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 487–494.

Kohavi, R., Becker, B. and Sommerfield, D.: 1997, Improving simple bayes, *Proceedings of the 9th European Conference on Machine Learning (ECML 97)*, Springer Verlag.

Kolcz, A. and Alspector, J.: 2001, Svm-based filtering of email spam with content-specific misclassification costs, *TextDM'2001 (IEEE ICDM-2001 Workshop on Text Mining)*, IEEE, pp. 123–130.

Kolodner, J.: 1992, An introduction to case-based reasoning, *AI Review* **6**(1), 3–34.

Kolter, J. and Maloof, M.: 2003, Dynamic weighted majority: a new ensemble method for tracking concept drift, *3rd IEEE International Conference on Data Mining*, IEEE CS Press, pp. 123–130.

Kubat, M. and Widmer, G.: 1995, Adapting to drift in continuous domains, *8th Eur Conf on Machine Learning*, Vol. 912 of *Lecture Notes in Computer Science*, Springer, pp. 307–310.

Kuncheva, L. I.: 2004, Classifier ensembles for changing environments., *in* F. Roli, J. Kittler and T. Windeatt (eds), *5th International Workshop on Multiple Classifier Systems (MCS 2004)*, Springer, pp. 1–15.

Landauer, T., Foltz, P. and Laham, D.: 1998, An introduction to latent semantic analysis, *Discourse Processes* **25**, 259–284.

Lenat, D., Davis, R., Doyle, J., Genesereth, M., Goldstein, I. and Schrobe, H.: 1983, Reasoning about reasoning, *in* F. Hayes-Roth, D. A. Waterman and D. B. Lenat (eds), *Building Expert Systems*, Addison-Wesley, London, pp. 219–239.

Lenz, M.: 1998, Defining knowledge layers for textual case-based reasoning, *EWCBR '98: Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag, London, UK, pp. 298–309.

Lenz, M., Auriol, E. and Manago, M.: 1998a, Diagnosis and decision support, *in* M. Lenz, B. Bartsch-Sporl, H. Burkhard and S. Wess (eds), *Case-Based Reasoning Technology, From Foundations to Applications*, LNCS, Springer-Verlag, pp. 51–90.

Lenz, M., Hubner, A. and Kunze, M.: 1998b, Textual CBR, *Case-Based Reasoning Technology, From Foundations to Applications*, Springer-Verlag, London, UK, pp. 115–138.

Lewis, D.: 1998, Naïve (bayes) at forty:the independence assumption in information retrieval, *Procs of 10th European Conference on Machine Learning (ECML 98)*, Vol. 1398 of *LNCS*, Springer, pp. 4–15.

Lewis, D. and Ringuette, M.: 1994, Comparison of two learning algorithms for text categorisation, *Procs of 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR 94)*, pp. 81–93.

Littlestone, N. and Warmuth, M.: 1994, The weighted majority algorithm, *Information and Computation* **108**(2), 212–261.

McKenna, E. and Smyth, B.: 2000, Competence-guided editing methods for lazy learning, *in* W. Horn (ed.), *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence*, IOS Press, pp. 60–64.

McLaren, B. M. and Ashley, K. D.: 2001, Helping a cbr program know what it knows, *in* D. Aha and I. Watson (eds), *4th International Conference on Case-Based Reasoning (ICCBR-2001)*, Vol. 2080 of *LNAI*, Springer, pp. 377–391.

Michelakis, E., Androutsopoulos, I., Paliouras, G., Sakkis, G. and Stamatopoulos, P.: 2004, Filtron: A learning-based anti-spam filter, *1st Conference on Email and Anti-Spam (CEAS 2004)*.

Mitchell, T.: 1997, *Machine Learning*, McGraw Hill, New York.

Mladenic, D.: 1998, Feature subset selection in text learning, *10th European conference on Machine Learning (ECML98)*, Vol. 1398 of *Lecture Notes in Computer Science*, Springer, pp. 95–98.

Nebel, B. and Koehler, J.: 1995, Plan reuse versus plan generation: a theoretical and empirical analysis, *Artificial Intelligence (Special Issue on Planning and Scheduling)* **76**(I–2), 427–454.

Niblett, T.: 1987, Constructing decision trees in noisy domains, *in* I. Bratko and N. Lavrac (eds), *Progress in Machine Learning, Procs of 2nd European Working Session on Learning (EWSL 87)*, Sigma Press, pp. 67–78.

Nugent, C., Cunningham, P. and Doyle, D.: 2005, The best way to instill confidence is by being right; and evaluation of the effectiveness of case-based explanations in providing user confidence, *in* H. Munoz-Avila and F. Ricci (eds), *6th International Conference on Case-Based Reasoning (ICCBR '05)*, Vol. 3620 of *LNAI*, Springer, pp. 368–381.

O'Brien, C. and Vogel, C.: 2003, Spam filters: Bayes vs. chi-squared; letters v. words, *International Symposium on Information and Communications Technology* .

Oda, T. and White, T.: 2003a, Developing an immunity to spam, *Procs of the Genetic and Evolutionary Computation Conference (GECC 03)*, Vol. 2723 of *Lecture Notes in Computer Science*, Springer, pp. 231–242.

Oda, T. and White, T.: 2003b, Increasing the accuracy of a spam-detecting artifical immune system, *Procs of the COngress on Evolutionary computation (CEC 03)*, Vol. 2723 of *Lecture Notes in Computer Science*, IEEE Computer Society.

Pantel, P. and Lin, D.: 1998, Spamcop: A spam classification and organisation program, *Procs of Workshop for Text Categorisation, AAAI-98*, pp. 95–98.

Plaza, E. and Ontanon, S.: 2003, Cooperative multiagent learning, **2636**, 1–17.

Provost, J.: 1999, Naïve bayes vs. rule-learning in classification of email, *Technical Report 99-284*, Department of Computer Science, University of Texas at Austin.

Quinlan, J. R.: 1997, *C4.5 Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Mateo, CA.

Reilly, J., Smyth, B., McGinty, L. and McCarthy, K.: 2005, Critiquing with confidence., *in* H. Muñoz-Avila and F. Ricci (eds), *Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, (ICCBR'05) Proceedings*, Vol. 3620 of *LNCS*, Springer, pp. 436–450.

Rennie, J.: 2000, ifile: an application of machine learning to e-mail filtering, *Proceedings of the KDD-2000 Workshop on Text Mining, Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Riesbeck, C. K. and Shank, R. C.: 1989, *Inside Case-based Reasoning*, Lawrence Erlbaum Associates., Cambridge MA.

Ritter, G. L., Woodruff, H. B., Lowry, S. R. and Isenhour, T. L.: 1975, An algorithm for a selective nearest neighbor decision rule, *IEEE Transactions on Information Theory* **21**(6), 665–669.

Rüping, S.: 2001, Incremental learning with support vector machines, *in* N. Cercone, T. Lin and X. Wu (eds), *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, IEEE, pp. 641–642.

Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E.: 1998, A bayesian approach to filtering junk email, *Procs of Workshop for Text Categorisation, AAAI-98*, pp. 55–62.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyroponlos, C. and Stamatopoulos, P.: 2001, Stacking classifiers for anti-spam filtering of email, *in* L. Lee and D. Harman (eds), *6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pp. 44–50.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. and Stamatopoulos, P.: 2003, A memory-based approach to anti-spam filtering for mailing lists, *Information Retrieval* **6**(1), 49–73.

Salganicoff, M.: 1997, Tolerating concept and sampling shift in lazy learning using prediction error context switching, *AI Review* **11**(1-5), 133–155.

Salzberg, S.: 1997, On comparing classifiers: pitfalls to avoid and a recommended approach, *Data Mining and Knowledge Discovery* **1**, 317–327.

Schapire, R. E.: 2002, The boosting approach to machine learning: An overview, *Proceedings of the MSRI workshop on nonlinear estimation and classification.*

Schneider, K.: 2003, A comparison of event models for näive bayes anti-spam e-mail filtering, *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pp. 307–314.

Sebastiani, F.: 2002, Machine learning in automated text categorization, *ACM Computing Surveys* **34**(1), 1–47.

Simon, H.: 1983, Why should machines learn?, *Machine Learning: an artificial intelligence approach* **1**, 392–399.

Smyth, B. and Cunningham, P.: 1996, The utility problem analysed, a case-base reasoning perspective, *in* I. Smith and B. Faltings (eds), *Advances in Case-Base Reasoning, Proceedings of EWCBR '96*, number 1168 in *LNAI*, Springer Verlag, pp. 392–399.

Smyth, B. and Keane, M.: 1995, Remembering to forget: A competence preserving case deletion policy for cbr system, *in* C. Mellish (ed.), *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI (1995)*, Morgan Kaufmann, pp. 337–382.

Smyth, B. and McKenna, E.: 1998, Modelling the competence of case-bases, *in* B. Smyth and P. Cunningham (eds), *Advances in Case-Based Reasoning, Proceedings of 4th European Workshop, EWCBR-98*, Vol. 1488 of *Lecture Notes in Computer Science*, Springer, pp. 208–220.

Stanley, K.: 2003, Learning concept drift with a committee of decision trees, *Technical Report UT-AI-TR-03-302*, Department of Computer Science, University of Texas at Austin.

Street, W. and Kim, Y.: 2001, A streaming ensemble algorithm (sea) for large-scale classification, *7th ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 377–382.

Syed, N. A., Liu, H. and Sung, K. K.: 1999, Handling concept drifts in incremental learning with support vector machines, *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 317–321.

Tomek, I.: 1976, An experiment with the nearest neighbor rule, *IEEE Transactions on Information Theory* **6**(6), 448–452.

Tsymbal, A.: 2004, The problem of concept drift: definitions and related work, *Technical Report TCD-CS-2004-15*, Department of Computer Science, Trinity College Dublin.

Tsymbal, A. and Puuronen, S.: 2000, Bagging and boosting with dynamic integration of classifiers., *in* D. A. Zighed, H. J. Komorowski and J. M. Zytkow (eds), *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2000)*, Springer, pp. 116–125.

Vapnik, V.: 1999, *The Nature of Statistical Learning Theory, 2nd. Ed.*, Statistics for Engineering and Information Science, Springer, New York.

Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E. and Blythe, J.: 1993, Derivational analogy in prodigy: Automating case acquisition, storage and utilisation, *Machine Learning* **10**, 249–278.

von Ahn, L., Blum, M. and Langford, J.: 2004, Telling humans and computers apart automatically, *CACM* **47**(1).

Wang, H., Fan, W., Yu, P. and Han, J.: 2003, Mining concept-drifting datastreams using ensemble classifiers, *9th ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 226–235.

Watson, I.: 1997, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann.

Watson, I. and Marir, F.: 1994, Case-based reasoning: a review, *The Knowledge Engineering Review* **9**(4), 327–354.

Weiss, G. and Provost, F.: 2003, Learning when training data are costly: The effect of class distribution on tree induction, *Journal of Artificial Intelligence Research* **19**, 315–354.

Wess, S., Althoff, K.-D. and Derwand, G.: 1994, Using k-d-trees to improve the retrieval step in case-base reasoning, *in* S. Wess, K.-D. Althoff and M. Richter (eds), *Topics in Case-Base Reasoning*, Vol. 837 of *LNAI*, Springer Verlag, pp. 167–181.

Widmer, G. and Kubat, M.: 1993, Effective learning in dynamic environments by explicit context tracking, *European Conference on Machine Learning (ECML 93)*, Vol. 667, Springer-Verlag, pp. 227–243.

Widmer, G. and Kubat, M.: 1996, Learning in the presence of concept drift and hidden contexts, *Machine Learning* **23**(1), 69–101.

Wilke, W. and Bergmann, R.: 1998, Techniques and knowledge used for adaptation during case-based problem solving, *IEA/AIE (Vol. 2)*, pp. 497–506.

Wilke, W., Smyth, B. and Cunningham, P.: 1998, Using configuration techniques for adaptation, *Case-Based Reasoning Technology, From Foundations to Applications*, Springer-Verlag, London, UK, pp. 139–168.

Wilson, D. L.: 1972, Asymptotic properties of nearest neighbor rules using edited data, *IEEE Transactions on Systems, Man and Cybernetics* **2**(3), 408–421.

Wilson, D. and Martinez, T.: 1997, Instance pruning techniques, *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., pp. 403–411.

Yang, Y. and Pedersen, J.: 1997, A comparative study on feature selection in text categorization, *ICML '97: Proceedings of the 14th International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., pp. 412–420.

Zamolotskikh, A., Delany, S. and Cunningham, P.: 2006, A methodology for comparing classifiers that allow the control of bias, *Proceedings of 21st ACM Symposium on Applied Computing*, ACM, pp. 582–587.

Zhang, J.: 1992, Selecting typical instances in instance-based learning, *Proceedings of the 9th International Conference on Machine Learning (ICML 92)*, Morgan Kaufmann Publishers Inc., pp. 470–479.

Zhang, L., Zhu, J. and Yao, T.: 2004, An evaluation of statistical spam filtering techniques, *ACM Transactions on Asian Language Information Processing (TALIP)* **3**(4), 243–269.

Zu, J. and Yang, Q.: 1997, Remembering to add: competence preserving case-addition policies for case-base maintenance, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 97)*, Morgan Kaufmann Publishers Inc., pp. 234–239.