ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΙΕΡΕΥΝΗΣΗ ΚΑΙ ΣΥΓΚΡΙΣΗ ΕΥΦΥΩΝ ΤΕΧΝΙΚΩΝ ΓΙΑ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ ΜΗΝΥΜΑΤΩΝ ΗΛΕΚΤΡΟΝΙΚΟΥ ΤΑΧΥΔΡΟΜΕΙΟΥ

ΣΥΓΓΡΑΦΕΑΣ: ΚΑΛΥΒΑΣ ΑΝΔΡΕΑΣ
ΚΑΘΗΓΗΤΗΣ: ΜΑΡΓΑΡΙΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
ΗΜΕΡΟΜΗΝΙΑ: 9-2-2006

1.Εισαγωγή

1.1 Τι είναι το Spam

Η λέξη Spam μπορεί να οριστεί επίσημα ως «Μη επιθυμητό ηλεκτρονικό μήνυμα που αποστέλλεται μαζικά»[1],[2],[5],[6]. Στα spam δεν συμπεριλαμβάνονται οι λίστες ηλεκτρονικού ταχυδρομείου τα newsletters και γενικά μηνύματα τα οποία ρητά έχει ζητήσει ο χρήστης ότι επιθυμεί να λαμβάνει. Συνώνυμος όρος είναι και τα αρχικά UCE (Unsolicited Commercial Email) που μεταφράζεται ως αυθαίρετο εμπορικό μήνυμα ηλεκτρονικού ταχυδρομείου. Κατά αντιστοιχία με τον όρο spam, έχει γίνει γενικά αποδεκτός ο όρος ham για την περιγραφή του συνόλου των επιθυμητών μηνυμάτων ηλεκτρονικού ταχυδρομείου.

1.2 Διεθνής αντιμετώπιση του Spam

Τα μηνύματα spam από έρευνες που έχουν γίνει φαίνεται να αποτελούν ένα ποσοστό 30-60% των μηνυμάτων που διακινούνται παγκοσμίως, και έχουν αυξητικές τάσεις[12]. Ο όγκος αυτός των μηνυμάτων αναγκάζει τους ιδιοκτήτες των εξυπηρετητών ηλεκτρονικών μηνυμάτων να αυξάνουν τις δαπάνες τους για αγορά εξοπλισμού λόγω κατασπατάλησης πόρων. Με ένα τόσο μεγάλο πλήθος ανεπιθύμητων μηνυμάτων οι χρήστες αναγκάζονται να σπαταλούν ένα μεγάλο μέρος του χρόνου τους για να τα διαχωρίζουν σε επιθυμητά και μη, κάτι που κοστίζει οικονομικά και στους εργοδότες τους. Επίσης στην πλειοψηφία των περιπτώσεων είναι αδύνατο να πετύχει κάποιος την αφαίρεση του από την εκάστοτε λίστα των παραληπτών των μηνυμάτων, παρά την παραπλανητική επιλογή που εμφανίζεται στο τέλος ενός μηνύματος spam.

Μερικές μέθοδοι που έχουν εφαρμοστεί για την αντιμετώπιση του φαινόμενου είναι οι εξής[4]:

- Η έκταση αυτή του προβλήματος έχει αναγκάσει εταιρίες αλλά και ανεξάρτητους πολίτες να στραφούν δικαστικά εναντίων των αποστολέων spam[13]. Σε κάποιες χώρες όπως η ΗΠΑ κάποιοι έχουν καταφέρει να κερδίσουν κάποιες δικαστικές αποφάσεις. Δυστυχώς όμως στις περισσότερες χώρες του κόσμου δεν έχει προχωρήσει η δικαστική αντιμετώπιση.

- Άλλη λύση που έχει εφαρμοστεί είναι η δημιουργία λιστών με τους αποστολείς των spam, από ομάδες παραληπτών που επιθυμούν να αρνούνται την παραλαβή μηνυμάτων από τέτοιους «σεσημασμένους» αποστολείς[12],[14]. Η λύση αυτή αν και καταφέρνει να κόψει μικρό αριθμό μηνυμάτων, δεν μπορεί να δώσει ουσιαστική λύση στο πρόβλημα λόγω των ιδιομορφιών του πρωτοκόλλου των μηνυμάτων ηλεκτρονικού ταχυδρομείου που αναφέρονται στην συνέχεια, και άρα είναι περιορισμένης αποτελεσματικότητας.
- Μια τρίτη κατηγορία λύσεων αφορά την προσπάθεια εφαρμογής αυτοματοποιημένων κανόνων για το φιλτράρισμα και διαχωρισμό των επιθυμητών από τα μη μηνυμάτων[7]. Στην πρώτη της μορφή η λύση εφαρμόστηκε με τον ορισμό κανόνων μέσα από τα προγράμματα ηλεκτρονικής αλληλογραφίας κυρίως από τους ίδιους τους χρήστες με χρήση λέξεων κλειδιών για τα μηνύματα. Και η μέθοδος αυτή είναι περιορισμένης αποτελεσματικότητας αφενός γιατί μπορεί να κοπούν επιθυμητά μηνύματα, αφετέρου γιατί οι αποστολείς των μη επιθυμητών μηνυμάτων μπορούν να την παρακάμψουν εύκολα (π.χ. αν κάποιος χρησιμοποιήσει την λέξη sex σαν κριτήριο απόρριψης ενός μηνύματος τότε μπορεί να κόψει κάποιο επιθυμητό μήνυμα που περιέχει την λέξη αυτή, αλλά και θα παραβλέψει τα ανεπιθύμητα μηνύματα που έχουν την παραποιημένη λέξη $ex)

1.3 Γιατί είναι δύσκολη η αντιμετώπιση του Spam

Το ηλεκτρονικό ταχυδρομείο είναι μια από τις πρώτες υπηρεσίες του διαδικτύου. Όταν τέθηκε σε λειτουργία χρήστες του ήταν μόνο η επιστημονική κοινότητα, και δεν υπήρχε εμπορική εκμετάλλευσή του. Για τον λόγο αυτό το πρωτόκολλο του ηλεκτρονικού ταχυδρομείου, που χρησιμοποιείται ίδιο μέχρι σήμερα, δεν δίνει κάποια μέθοδο ασφάλειας των δεδομένων ή ταυτοποίησης του αποστολέα. Άμεσο αποτέλεσμα είναι ότι σε κάθε ηλεκτρονικό μήνυμα είναι αδύνατη η σίγουρη εξακρίβωση της ταυτότητας του αρχικού αποστολέα, αφού αυτή μπορεί να παραποιηθεί. Το γεγονός αυτό εκμεταλλεύονται και οι αποστολείς των ανεπιθύμητων μηνυμάτων έτσι ώστε αφ' ενός να μην γίνεται δυνατή η εύρεση των πραγματικών στοιχείων τους, αφ' ετέρου να μην είναι εφικτό το φιλτράρισμα και η απόρριψη των μηνυμάτων τους με την χρήση των απλών λιστών «σεσημασμένων» αποστολέων που αναφέρθηκαν προηγουμένως.

Τα προγράμματα ηλεκτρονικού ταχυδρομείου προσφέρουν συνήθως την δυνατότητα εφαρμογής φίλτρων στα εισερχόμενα μηνύματα βάση διαφόρων παραγόντων (π.χ. αποστολέας, θέμα, παραλήπτης κτλ). Η δυνατότητα αυτή δουλεύει πολύ καλά στην περίπτωση διαχωρισμού των επιθυμητών μηνυμάτων σε διάφορες κατηγορίες ανάλογα με τον αποστολέα ή το θέμα στην επικεφαλίδα του μηνύματος αλλά είναι ανεπαρκής στην περίπτωση του spam, όπου ο αποστολέας κακόβουλα και σκόπιμα αποκρύπτει την πραγματική του ταυτότητα και παραπλανητικά αναφέρει ψευδή θέματα στην επικεφαλίδα.

Επίσης η οδός της δικαστικής διαμάχης μπορεί να πέτυχε σε λίγες περιπτώσεις, αλλά αυτό έγινε μόνο για μεμονωμένα περιστατικά σε χώρες όπως οι Η.Π.Α., που έχουν το κατάλληλο νομικό πλαίσιο, και χωρίς να επιτευχθεί κάποιο σημαντικό αποτέλεσμα ενώ ταυτόχρονα η παγκοσμιοποίηση του διαδικτύου και η δυνατότητα απόκρυψης του αρχικού αποστολέα των μηνυμάτων δεν διευκολύνουν την συλλογή ενοχοποιητικών στοιχείων[4]. Το ζήτημα που παραμένει ανοιχτό στη δικαστική αντιμετώπιση είναι το ποιο δίκαιο θα εφαρμοστεί, αυτό της χώρας του αποστολέα ή του παραλήπτη;

Όλα τα παραπάνω δείχνουν ότι οι παλαιότερες μέθοδοι δεν είναι αποτελεσματικές και μάλιστα όταν εφαρμόζονται μεμονωμένα η κάθε μία[4]. Νέες μέθοδοι πρέπει να εφαρμοστούν στο αυτοματοποιημένο φιλτράρισμα των μηνυμάτων με βάση όχι τις πληροφορίες στην επικεφαλίδα ενός μηνύματος (π.χ. αποστολέας, προέλευση, θέμα κτλ) αλλά το ίδιο το περιεχόμενο του.

Λόγω της σοβαρότητας του προβλήματος αλλά και της αυξητικής του τάσης έχουν ήδη αρχίσει να προτείνονται νέες λύσεις που βασίζονται στο περιεχόμενο των μηνυμάτων, αποδίδουν καλύτερα από τις παλαιότερες μεθόδους που αναφέρθηκαν προηγουμένως και φέρνουν καλύτερα αποτελέσματα[4]. Μια από τις λύσεις αυτές είναι και αυτή που εξετάζεται στην παρούσα εργασία και βασίζεται στην χρήση μεθόδων Bayes ώστε κάθε χρήστης να μπορεί να «εκπαιδεύσει» κάποιον αλγόριθμο με επιθυμητά και ανεπιθύμητα μηνύματα ηλεκτρονικού ταχυδρομείου και στην συνέχεια ο αλγόριθμος να μπορεί να διαχωρίσει τα μηνύματα σε επιθυμητά και ανεπιθύμητα.

1.4 Σκοπός της εργασίας

Πρόσφατα, λόγω της έξαρσης του φαινόμενου του spam, έχουν εμφανιστεί πλήθος μεθόδων λογισμικού, βασισμένες σε θεωρία Bayes, για την αντιμετώπιση των ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου. Ωστόσο υπάρχει έλλειψη μεθόδων, αλλά και αποτελεσμάτων, αξιολόγησης του λογισμικού αυτού για την αντιμετώπιση του spam. Το κενό αυτό έρχεται να καλύψει η εργασία αυτή.

Πράγματι αν κοιτάξει κάποιος τις περιγραφές των προγραμμάτων όπως παρουσιάζονται από τις εταιρίες ανάπτυξης τους θα καταλήξει στο συμπέρασμα ότι όλα αυτά τα προγράμματα καταφέρνουν πλήρως και 100% να ξεχωρίσουν τα μη επιθυμητά μηνύματα. Τα αποτελέσματα όμως σε καθημερινή χρήση των προγραμμάτων δείχνουν ότι η κατάσταση απέχει αρκετά από το ιδανικό, όπως αυτό παρουσιάζεται. Επιπλέον πολλά από τα προγράμματα μπορεί να επιτελούν πολύ καλό έργο την πρώτη στιγμή της κυκλοφορίας τους αλλά μετά από μερικούς μήνες να χάνουν μεγάλο ποσοστό της απόδοσης τους γιατί οι διάφοροι αποστολείς ανεπιθύμητων μηνυμάτων βρίσκουν τρόπους για να τα «ξεγελάνε». Επίσης σημαντικό ρόλο στην απόδοση ενός αλγορίθμου παίζει και ο τρόπος που γίνεται η εκπαίδευση. Πράγματι όπως θα δείξουμε στην συνέχεια της εργασίας σημαντικότατο ρόλο στην απόδοση των προγραμμάτων παίζει το αν η εκπαίδευση γίνει με κάποια γενικά μηνύματα ηλεκτρονικού ταχυδρομείου, τα οποία υπάρχουν διαθέσιμα γι' αυτόν τον σκοπό σε διάφορες διευθύνσεις στο internet, ή αν χρησιμοποιήθηκαν προσωπικά δεδομένα του χρήστη.

Στην αξιολόγηση παίζει σημαντικό ρόλο και το είδος του σφάλματος που εμφανίζεται. Υπάρχουν δυο είδη σφάλματος. Το ένα αφορά ανεπιθύμητα μηνύματα που κατατάσσονται ως επιθυμητά και το οποίο θα ονομάσουμε False Negative και το άλλο αφορά επιθυμητά μηνύματα που κατατάσσονται ως ανεπιθύμητα, το οποίο θα ονομάσουμε False Positive. Ενώ για πολλούς είναι αποδεκτό ένα μικρό σφάλμα False Negative δηλαδή να καταταγούν μερικά ανεπιθύμητα μηνύματα ως επιθυμητά, για τον περισσότερο κόσμο είναι πολύ λιγότερο αποδεκτό από κάποιο λογισμικό να

εμφανίζει σφάλματα False Positive, δηλαδή να κατατάσσονται επιθυμητά μηνύματα ως spam.

Το κενό αυτό της αξιολόγησης που υπάρχει σήμερα προσπαθεί να καλύψει η εργασία αυτή. Γίνονται αντικειμενικές μετρήσεις της απόδοσης των προγραμμάτων σε ένα σύνολο αποδεκτών και μη αποδεκτών μηνυμάτων και καταγράφεται με ακρίβεια η απόδοση των διαφόρων προγραμμάτων. Η επιλογή των μετρήσεων που χρησιμοποιήθηκαν έγινε με τέτοιο τρόπο ώστε να καλυφθούν όλοι οι παράγοντες που αναφέρθηκαν προηγουμένως και οι οποίοι μπορούν να επηρεάσουν το αποτέλεσμα.

1.5 Δομή εργασίας

Το πρώτο κεφάλαιο είναι μια εισαγωγή στο πρόβλημα του spam όπου αναφέρουμε την σημερινή κατάσταση καθώς και τους λόγους για τους οποίους είναι δύσκολη η αντιμετώπιση του φαινόμενου αυτού. Επιπλέον θέτονται οι στόχοι της εργασίας και περιγράφεται η μεθοδολογία που ακολουθήθηκε.

Το δεύτερο κεφάλαιο της εργασίας ασχολείται με μια θεωρητική εισαγωγή στην αντιμετώπιση του φαινόμενου του spam. Γίνεται αρχικά μια αναφορά σε διάφορες εργασίες που έχουν γίνει γύρω από το θέμα της αντιμετώπισης ανεπιθύμητων μηνυμάτων. Στην συνέχεια παρατίθεται μια σύντομη περιγραφή της στατιστικής Bayes πάνω στις αρχές της οποίας βασίζεται η λειτουργία των προγραμμάτων που εξετάζουμε στην εργασία αυτή.

Στο τρίτο κεφάλαιο περιγράφουμε όλες τις παραμέτρους των μετρήσεων που πραγματοποιήθηκαν. Αρχικά περιγράφουμε το λογισμικό και το υλικό του συστήματος στο οποίο έγιναν οι μετρήσεις. Στην συνέχεια αναφερόμαστε στα corpus, σύνολα δηλαδή από μηνύματα ηλεκτρονικού ταχυδρομείου, επιθυμητά και ανεπιθύμητα, που χρησιμοποιήθηκαν στις δοκιμές. Ακολουθεί μια ανάλυση της δομής ενός επιθυμητού αλλά και ενός ανεπιθύμητου μηνύματος, ώστε ο αναγνώστης να εξοικειωθεί με τις πληροφορίες που χρησιμοποιούν τα προγράμματα για να κατατάξουν τα μηνύματα. Μετά γίνεται μια γενική περιγραφή των προγραμμάτων που χρησιμοποιήθηκαν. Τα προγράμματα είναι τα εξής:

- Spamassassin

- Dspam
- Spambayes
- Spamprobe
- Bogofilter

Τέλος περιγράφουμε τις ίδιες τις μετρήσεις που πραγματοποιήθηκαν. Περιληπτικά οι μετρήσεις αυτές είναι:

- Εκπαίδευση με ένα γενικό corpus και έλεγχος της απόδοσης με ένα άλλο γενικό corpus. Με τον όρο γενικό corpus εννοούμε ένα corpus που δίνεται ελεύθερα κάπου στο διαδίκτυο. Στην περίπτωση αυτής της εργασίας χρησιμοποιήθηκαν corpus που προσφέρονται από την σελίδα του Spamassassin. Τα corpus αυτά έχουν χρησιμοποιηθεί και σε άλλες αντίστοιχες εργασίες και αποτελούν ένα γενικά αποδεκτό, κοινό μέτρο σύγκρισης των προγραμμάτων διαχωρισμού των spam.
- Εκπαίδευση με ένα γενικό corpus και έλεγχος με ένα corpus από δεδομένα που προέρχονται από προσωπικά δεδομένα του γράφοντος.
- Έλεγχος της μεταβολής της απόδοσης των προγραμμάτων ανάλογα με το μέγεθος του dataset εκμάθησης.

Στο τέταρτο κεφάλαιο παρουσιάζονται με την μορφή γραφημάτων τα αναλυτικά αποτελέσματα για κάθε μια από τις μετρήσεις που πραγματοποιήθηκαν και για κάθε ένα από τα προγράμματα χωριστά. Τα γραφήματα συνοδεύονται από εκτενή σχολιασμό των αποτελεσμάτων.

Στο πέμπτο κεφάλαιο γίνεται μια σύνοψη των αποτελεσμάτων με συγκεντρωτικά διαγράμματα για κάθε μέτρηση ώστε να συγκριθεί η απόδοση των προγραμμάτων μεταξύ τους και να προκύψουν κάποια γενικά συμπεράσματα.

Ακολουθεί η βιβλιογραφία καθώς και ένα παράρτημα με τον κώδικα που αναπτύχθηκε για την διεξαγωγή των μετρήσεων.

1.6 Σύνοψη αποτελεσμάτων

Αναλυτικά τα αποτελέσματα θα παρουσιαστούν σε επόμενα κεφάλαια. Στην συνέχεια αναφέρουμε σε γενικές γραμμές τα αποτελέσματα και τα συμπεράσματα στα οποία κατέληξε η εργασία ώστε να αποκτήσει ο αναγνώστης μια εικόνα της απόδοσης των προγραμμάτων που εξετάστηκαν.

Στην πρώτη μέτρηση όλα γενικά τα προγράμματα απέδωσαν καλά και μπόρεσαν να διαχωρίσουν σε μεγάλο βαθμό τα επιθυμητά από τα ανεπιθύμητα μηνύματα. Ωστόσο διαφορές στην απόδοσή τους ήταν υπαρκτές και θα αναφερθούμε σε αυτές αναλυτικότερα σε επόμενα κεφάλαια.

Στην δεύτερη μέτρηση οι αποδόσεις των προγραμμάτων ήταν γενικά μειωμένες σε μεγάλο βαθμό, σε σχέση με την πρώτη μέτρηση, και μάλιστα σε μερικές περιπτώσεις, κυρίως του Bogofilter, μη αποδεκτές. Από τα αποτελέσματα αυτά μπορούμε να συμπεράνουμε ότι για την μέγιστη απόδοση των προγραμμάτων πρέπει να πραγματοποιείται εκπαίδευση με μηνύματα των ιδίων των χρηστών που θα τα χρησιμοποιήσουν και όχι με κάποιο γενικό corpus.

Από την τρίτη μέτρηση φαίνεται ότι όλα γενικά τα προγράμματα είτε σε μικρό είτε σε μεγαλύτερο βαθμό ευνοούνται από την χρήση όσο το δυνατόν μεγαλύτερου corpus εκμάθησης για την εκπαίδευση τους. Η επίδραση του πλήθους των μηνυμάτων εκπαίδευσης είναι κυρίως αισθητή στον διαχωρισμό των ανεπιθύμητων μηνυμάτων αφού η απόδοση των προγραμμάτων στον διαχωρισμό τους αυξάνεται σε μεγάλο βαθμό όταν αυξάνεται και το dataset εκμάθησης.

Αν προσπαθήσουμε να κατατάξουμε τα προγράμματα τους με βάση τις τρεις μετρήσεις που πραγματοποιήθηκαν, μάλλον την βέλτιστη απόδοση μεταξύ τους παρουσιάζει το Spambayes. Το πρόγραμμα αυτό παρουσίασε, με αρκετή διαφορά από τα υπόλοιπα, την καλύτερη απόδοση στις μετρήσεις 1 και 3. Είναι λοιπόν αυτό που θα επιδείξει την καλύτερη συμπεριφορά σε πραγματικές συνθήκες λειτουργίας. Μοναδικό σημείο στο οποίο συνίσταται προσοχή είναι η εκπαίδευσή του με δεδομένα των μηνυμάτων που δέχονται οι χρήστες και όχι κάποιο γενικό dataset από το Internet, καθώς όπως φαίνεται από την μέτρηση 2 η απόδοσή του μειώνεται σε

μεγάλο βαθμό όταν τα δεδομένα εκπαίδευσης δεν είναι αρκετά συσχετισμένα με τα δεδομένα ελέγχου. Αντίστοιχη μεγάλη μείωση της απόδοσης βεβαίως συναντούμε και στα υπόλοιπα προγράμματα.

Στην δεύτερη θέση έρχεται το Spamassassin καθώς παρουσιάζει την δεύτερη καλύτερη απόδοση στις μετρήσεις 1 και 3, μετά το Spambayes, και την καλύτερη απόδοση στην μέτρηση 2. Το Spamassassin είναι λοιπόν μια καλή επιλογή αφού εμφανίζει πολύ καλή, αλλά όχι άριστη, απόδοση σε όλους τους τομείς και δεν μειονεκτεί σε κανέναν.

Ακολουθούν τα Spamprobe, Dspam τα οποία εμφανίζουν προβλήματα σε διαφορετικούς τομείς. Το Spamprobe τα πήγε πολύ καλά στην μέτρηση με το πλήρες corpus αλλά χάνει μεγάλο μέρος της απόδοσης του όταν γίνεται εκπαίδευση με μικρό corpus. Επίσης άσχημη είναι η απόδοσή του και στην μέτρηση 2. Το σημαντικότερο μειονέκτημα του Dspam είναι ότι εμφανίζει αρκετά μεγάλο σφάλμα False Positive κάτι που το κάνει όχι και τόσο ελκυστικό στα μάτια των περισσοτέρων χρηστών.

Τέλος την χειρότερη επίδοση από όλα τα προγράμματα σε όλες τις μετρήσεις εμφάνισε το Bogofilter του οποίου η χρήση δεν συνίσταται.

2. Βιβλιογραφία - Εφαρμογές

2.1 Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένας τομέας με ευρεία και πολύχρονη ιστορία. Γενικά ένας αλγόριθμος μηχανικής μάθησης έχει την ιδιότητα να μεταβάλει την συμπεριφορά του λαμβάνοντας υπ' όψη αποτελέσματα του παρελθόντος. Στην περίπτωση των προγραμμάτων διαχωρισμού του spam η μάθηση βασίζεται στην ύπαρξη δυο προδιαχωρισμένων corpuses ένα για spam και ένα για ham[1],[2],[3],[5]. Σύμφωνα με την μέθοδο αυτή ο αλγόριθμος επεξεργάζεται τα παραδείγματα εκμάθησης κατά την φάση της εκπαιδευσης (training phase). Οι κανόνες που προκύπτουν στην συνέχεια χρησιμοποιούνται χωρίς επιπλέον μεταβολές για την φάση του διαχωρισμού (classification phase). Η δυνατότητα διόρθωσης των αποτελεσμάτων κατά την διάρκεια της φάσης διαχωρισμού είναι επίσης πολύτιμη. Αυτό επιτρέπει στο σύστημα να προσαρμόζεται σε μεταβαλλόμενες καταστάσεις όπως σε προτιμήσεις χρηστών ή νέο περιεχόμενο ανεπιθύμητων μηνυμάτων. Η απλοϊκή μέθοδος επανεκπαίδευσης με ολόκληρο το corpus επαρκεί συνήθως, αν ο αλγόριθμος είναι αρκετά γρήγορος.

Η εκπαίδευση με επίβλεψη για τον διαχωρισμό των μηνυμάτων ξεκινά με ένα corpus που περιέχει μια συλλογή σωστά διαχωρισμένων μηνυμάτων spam και ham. Στο στάδιο της επιλογής χαρακτηριστικών, χαρακτηριστικά που ξεχωρίζουν αν ένα μήνυμα είναι spam ή ham αναγνωρίζονται. Στο στάδιο της εκπαίδευσης τα επιλεγμένα χαρακτηρίστηκα του corpus μελετώνται για να μαθευτούν γνωρίσματα που μπορούν να ξεχωρίσουν αν ένα μήνυμα είναι spam η ham. Στην συνέχεια ή και ταυτόχρονα ένα στάδιο επαλήθευσης χρησιμοποιείται για να ελέγξει την αποτελεσματικότητα των χαρακτηριστικών που μαθεύτηκαν. Τέλος η συσσωρευμένη γνώση χρησιμοποιείται σε ένα στάδιο διαχωρισμού που φιλτράρει ανεπιθύμητα μηνύματα κάνοντας έναν διαχωρισμό σε κάθε μήνυμα από ένα σύνολο μηνυμάτων που χρησιμοποιούνται για το στάδιο αυτό.

Ένα σημαντικό χαρακτηριστικό που πρέπει να λάβει κανείς υπ' όψη κατά την διάρκεια μιας διαδικασίας μάθησης με επίβλεψη αφορά την διαχείριση του corpus. Για καλύτερη ακρίβεια θα ήθελε κάποιος να χρησιμοποιήσει όλο το corpus σαν δεδομένα εκπαίδευσης. Δυστυχώς αυτό κάνει την επαλήθευση αρκετά δύσκολη. Ο διαχωριστής (classifier) θα φανεί ότι αποδίδει υπερβολικά καλά όταν κληθεί να διαχωρίσει τα μηνύματα από τα οποία έμαθε. Ευτυχώς στα πιο πολλά προβλήματα ο αριθμός των δειγμάτων από τα οποία χρειάζεται να μάθει ο αλγόριθμος για να πετύχει δεδομένη ακρίβεια αυξάνεται απλώς λογαριθμικά σε σχέση με τον χώρο των πιθανών γεγονότων.

2.2 Θεωρία Bayes

2.2.1 Εισαγωγή

Από φιλοσοφικής σκοπιάς υπάρχουν ριζικές διαφορές μεταξύ της κλασσικής στατιστικής και της στατιστικής Bayes.Το βασικό σημείο της θεωρίας Bayes είναι ότι η πιθανότητα είναι μια έκφραση του βαθμού εμπιστοσύνης, ενώ στην κλασσική στατιστική είναι ένα μέτρο σχετικής συχνότητας. Όσον αφορά τον τομέα των συστημάτων αυτόματης εκμάθησης και λήψης αποφάσεων η ερμηνεία της πιθανότητας από την θεωρία Bayes είναι η πλέον ενδεδειγμένη. Στην εκτίμηση των παραμέτρων η κλασσική μέθοδος θεωρεί την παράμετρο σαν μια άγνωστη σταθερά, ενώ η στατιστική Bayes σαν τυχαία μεταβλητή.

Η γνώση είναι πάντα υποκειμενική και γι' αυτό όλες οι πιθανότητες που εμφανίζονται στην θεωρία Bayes είναι σχετικές. Ειδικότερα μέσω αυτής της θεώρησης η πιθανότητα δεν είναι μια αντικειμενική ιδιότητα ενός φυσικού γεγονότος αλλά σχετίζεται με προηγούμενες υποθέσεις και την εμπειρία του συστήματος μάθησης. Είναι απόλυτα φυσιολογικό να μιλάμε για «την πιθανότητα να υπάρχει ένας δέκατος πλανήτης στο ηλιακό σύστημα», ωστόσο δεν έχει νόημα να θεωρήσουμε την πιθανότητα ως την συχνότητα παρατήρησης ενός δέκατου πλανήτη.

2.2.2 Προτάσεις

Στην θεωρία πιθανοτήτων Bayes, οι πιθανότητες ορίζονται για προτάσεις που ακολουθούν τους κανόνες της άλγεβρας Bool και μπορεί να είναι είτε αληθείς είτε ψευδείς. Θα ορίσουμε αυτές τις προτάσεις χρησιμοποιώντας κεφαλαία γράμματα. Τρεις βασικές λειτουργίες ορίζονται στην άλγεβρα Bool, η τομή, η ένωση και η αντίθεση. Από εδώ και στο εξής θα χρησιμοποιήσουμε τους εξής συμβολισμούς «*A* and *B*» ως *AB*, «*A* or *B*» ως *A*+*B* και «Not *A*» ως $\neg A$

Η φυσική γλώσσα περιλαμβάνει εκφράσεις που μπορεί να είναι διφορούμενες. Π.χ. «ο ουρανός είναι γαλάζιος». Ο ορισμός των λέξεων ουρανός και γαλάζιος είναι λίγο πολύ διφορούμενος και γι' αυτό μπορούμε να δεχτούμε ότι η αληθινή τιμή της έκφρασης «ο ουρανός είναι γαλάζιος» δεν είναι ούτε τελείως αληθής ούτε τελείως ψευδής αλλά κάτι ενδιάμεσο. Από εδώ και στο εξής θα ασχοληθούμε μόνο με σαφώς ορισμένες προτάσεις.

2.2.3 Βασικοί κανόνες της θεωρίας Bayes

Η θεωρία πιθανοτήτων Bayes μπορεί να περιγραφεί από τους ακόλουθους βασικούς κανόνες:

- Κανόνας αθροίσματος :
  $P(A \mid B) + P(\neg A \mid B) = 1$

- Κανόνας γινομένου :
  $P(AB \mid C) = P(A \mid C)P(B \mid AC)$

Εδώ το $P(A \mid B)$ δηλώνει την πιθανότητα του *A* υπό την προϋπόθεση ότι το *B* είναι αληθές.

Αυτοί οι κανόνες ρυθμίζουν την κλίμακα στην οποία οι βαθμοί εμπιστοσύνης εφαρμόζονται.

2.2.4 Ο κανόνας Bayes

Από τον κανόνα του γινομένου μπορεί να παραχθεί ο βασικός κανόνας του Bayes:

$$P(A \mid BC) = \frac{P(A \mid C)P(B \mid AC)}{P(B \mid C)}$$

Αν υποθέσουμε ότι το *A* είναι μια από τις πολλές εξηγήσεις για μια νέα παρατήρηση, ότι το *B* είναι η νέα παρατήρηση και ότι το C είναι οι προηγούμενες γνώσεις και εμπειρία, βλέπουμε ότι ο κανόνας του Bayes λέει πως το σύστημα μάθησης πρέπει να μεταβάλλει την γνώση του καθώς λαμβάνει μια καινούρια παρατήρηση.

Πριν κάνουμε την παρατήρηση B, το σύστημα μάθησης γνωρίζει μόνο το C, αλλά στην συνέχεια γνωρίζει το BC, δηλαδή γνωρίζει το «B and C». Ο κανόνας του Bayes στην συνέχεια περιγράφει πως το σύστημα μάθησης πρέπει να προσαρμοστεί (από P(A|C) σε P(A|BC)), ως απάντηση στην παρατήρηση. Για να έχει χρησιμότητα ο κανόνας, η εξήγηση A χρειάζεται να είναι τέτοια ώστε μαζί με τις προηγούμενες υποθέσεις και εμπειρία, C, να δίνει σωστή τιμή στην πιθανότητα P(B|AC).

Συνήθως η πιθανότητα P(A|C) ονομάζεται prior πιθανότητα ενώ η P(A|BC) ονομάζεται posterior πιθανότητα. Πρέπει να σημειωθεί εδώ ότι αυτή η διάκριση είναι σχετική με την παρατήρηση. Η posterior πιθανότητα μιας παρατήρησης είναι η prior πιθανότητα για την επόμενη παρατήρηση.

2.2.6 Θεωρία Αποφάσεων

Η γνώση απλώς των πεποιθήσεων (beliefs) δεν επαρκεί για την λήψη αποφάσεων. Χρειάζονται και προτιμήσεις (preferences). Η θεωρία αποφάσεων ορίζει πως πρέπει οι πιθανότητες και οι προτιμήσεις να συνδυάζονται για την λήψη αποφάσεων. Δηλώνουμε ως *U(A)* την χρησιμότητα (Utility) της πρότασης *A*. Το *A* είναι προτιμότερο του *B* αν *U(A)>U(B)*.

Ο βασικός κανόνας της θεωρίας αποφάσεων είναι ο εξής:

$$U(A) = P(B \mid A)U(AB) + P(\neg B \mid A)U(A\neg B)$$

Στην γενικότερη περίπτωση των ανεξάρτητων μεταξύ τους προτάσεων $B_1, B_2, \ldots$ παίρνει την μορφή:

$$U(A) = \sum_i P(B_i \mid A)U(AB_i)$$

Η σημασία του κανόνα γίνεται εμφανής αν κάποιος θεωρήσει το Α ως μια δράση και τα B_i τις πιθανές συνέπειες. Βασικά ο κανόνας δείχνει ότι η χρησιμότητα του Α εξαρτάται από τις χρησιμότητες των πιθανών συνεπειών του Α, σε συνδυασμό με τις πιθανότητες των συνεπειών.

Οι κανόνες αθροίσματος και γινομένου της θεωρίας πιθανοτήτων ορίζουν την κλίμακα με την οποία οι βαθμοί εμπιστοσύνης ορίζονται ως το μέτρο των πιθανοτήτων. Ο κανόνας της χρησιμότητας κάνει το ίδιο για τις χρησιμότητες.

2.3 Ένα σχέδιο για το Spam

Στο κείμενο αυτό[6] ο Paul Graham προτείνει μια βασική μέθοδο εφαρμογής της θεωρίας Bayes για τον διαχωρισμό μηνυμάτων ηλεκτρονικής αλληλογραφίας σε spam ή ham. Σύμφωνα με το άρθρο αυτό η αχίλλειος πτέρνα των μηνυμάτων spam είναι το ίδιο το περιεχόμενο του μηνύματος. Ο αποστολέας μπορεί να ξεπεράσει συνήθως άλλες μεθόδους προστασίας αλλά πρέπει οπωσδήποτε να παραδώσει το ίδιο το μήνυμα. Οπότε η λύση είναι η συγγραφή προγραμμάτων που θα αναγνωρίζουν το περιεχόμενο του μηνύματος.

Για τον παραλήπτη τα ανεπιθύμητα μηνύματα είναι εύκολα αναγνωρίσιμα. Τι χρειάζεται λοιπόν να γίνει για να αυτοματοποιηθεί η διαδικασία;

Στο άρθρο αυτό ο συγγραφέας προτείνει ότι το πρόβλημα μπορεί να λυθεί με σχετικά απλούς αλγορίθμους. Ένας τέτοιος είναι η χρήση του κανόνα Bayes για τον συνδυασμό των πιθανοτήτων, να είναι spam το μήνυμα, για κάθε μια από τις λέξεις

του μηνύματος. Συνήθως δεν σκέφτεται κάποιος εύκολα να χρησιμοποιήσει στατιστικούς κανόνες για την ανίχνευση ανεπιθύμητων μηνυμάτων αλλά χρησιμοποιεί κάποιους απλούς κανόνες όπως το αν υπάρχει η λέξη «click» στο κείμενο. Πραγματικά ο έλεγχος για αυτήν την λέξη θα ανιχνεύσει μεγάλο αριθμό από ανεπιθύμητα μηνύματα. Ωστόσο αυτή η μέθοδος δημιουργεί 2 σοβαρά προβλήματα:

- Η απόδοση της μειώνεται όσο προσπαθεί κάποιος να προσεγγίσει το 100% στην ανίχνευση των ανεπιθύμητων μηνυμάτων.
- Το πιο σημαντικό πρόβλημα είναι ότι δημιουργεί false positives.

Η μέθοδος που εφαρμόζεται για την ανίχνευση είναι η εξής. Αρχικά χρησιμοποιούνται δύο συλλογές μηνυμάτων (corpora) μία με spam και μία με ham. Για κάθε μήνυμα σε κάθε corpus ο αλγόριθμος σαρώνει όλο το περιεχόμενο συμπεριλαμβανομένων των headers, html, javascript κλπ. Οι αλφαριθμητικοί χαρακτήρες, οι απόστροφοι και το σύμβολο του δολαρίου θεωρούνται μέρος των συμβόλων ενώ όλοι οι άλλοι χαρακτήρες θεωρούνται διαχωριστικά των συμβόλων. Ο συγγραφέας αναφέρει ότι μπορεί τα αποτελέσματα να βελτιωθούν αν γίνει καλύτερος διαχωρισμός στο τι αντιπροσωπεύει κάθε χαρακτήρας.

Δημιουργούνται αρχικά δύο πίνακες. Ο ένας έχει την συχνότητα εμφάνισης των λέξεων που εμφανίζονται στα μηνύματα ham ενώ ο άλλος περιέχει την συχνότητα εμφάνισης των λέξεων που εμφανίζονται στα μηνύματα spam. Στην συνέχεια δημιουργείται ένας τρίτος πίνακας με τις πιθανότητες ένα μήνυμα να είναι spam όταν περιέχει κάθε μια από τις λέξεις που εμφανίζονται στους δυο άλλους πίνακες. Έχουμε:

$$P(w) = \frac{\frac{s(w)}{l(spam)}}{\frac{2 \cdot h(w)}{l(ham)} + \frac{s(w)}{l(spam)}}$$

Όπου w είναι μια λέξη και P(w) είναι η πιθανότητα ένα μήνυμα να είναι spam αν περιέχει αυτή τη λέξη. Το s(w) είναι ο αριθμός των spam στα οποία εμφανίζεται η λέξη w και το h(w) είναι ο αριθμός των ham στα οποία εμφανίζεται η λέξη w.Επίσης l(spam) είναι ο αριθμός των μηνυμάτων που περιέχεται στο corpus του spam ενώ l(ham) είναι ο αριθμός των μηνυμάτων που περιέχεται στο corpus του ham. Το

$2 \cdot h(w)$ χρησιμοποιείται για να γίνει ο αλγόριθμος λιγότερο ευαίσθητος στην ανίχνευση false positives. Όταν φθάνει ένα καινούριο μήνυμα τότε υπολογίζεται η συνδυασμένη πιθανότητα όλων των λέξεων του:

$$P = \frac{\prod_i P(w_i)}{\prod_i P(w_i) + \prod_i (1 - P(w_i))}$$

Η σχέση αυτή δίνει την πιθανότητα να είναι spam ένα μήνυμα που περιέχει τις λέξεις $w_1, w_2, \ldots$

2.4 Μια στατιστική προσέγγιση στο πρόβλημα του Spam

Στο άρθρο αυτό[15] που εμφανίζεται στο Linux Journal ο Gary Robinson προτείνει μια βελτιωμένη εκδοχή της μεθόδου που πρότεινε ο Paul Graham στο "A plan for spam"[6]. Το άρθρο αυτό βασίζεται στην μέθοδο chi-square για τον συνδυασμό των επιμέρους πιθανοτήτων των λέξεων ενός μηνύματος ηλεκτρονικού ταχυδρομείου.

Σύμφωνα με την μέθοδο αυτή για κάθε λέξη που εμφανίζεται στα corpus που διαθέτουμε υπολογίζουμε τις πιθανότητες:

- b(w) = (ο αριθμός των spam μυνημάτων που περιέχουν την λέξη w) / (ο συνολικός αριθμός των spam μηνυμάτων)
- g(w) = (ο αριθμός των ham μηνυμάτων που περιέχουν την λέξη w) / (ο συνολικός αριθμός των spam μηνυμάτων)
- $p(w) = \frac{b(w)}{b(w) + g(w)}$

Η πιθανότητα p(w) μπορεί να θεωρηθεί ως η πιθανότητα ένα τυχαίο μήνυμα που έχει την λέξη w να είναι spam

Χρησιμοποιώντας την θεωρία Bayes υπολογίζεται ο βαθμός εμπιστοσύνης f(w) για κάθε λέξη ως εξής:

$$f(w) = \frac{(s \cdot x) + (n \cdot p(w))}{s + n}$$

όπου:

- x είναι η πιθανότητα που έχει κάποια λέξη, την οποία δεν έχουμε συναντήσει ξανά, να βρίσκεται σε ένα μήνυμα spam
- s είναι η ισχύς που θέλουμε να δώσουμε στην εκ προτέρων γνώση x
- n είναι ο αριθμός των μηνυμάτων που περιέχουν την λέξη w

Σύμφωνα με τον συγγραφέα του άρθρου η χρήση της f(w) αντί της p(w) δίνει σχεδόν πάντα καλύτερα αποτελέσματα στον διαχωρισμό επιθυμητών και ανεπιθύμητων μηνυμάτων.

Η συνδιασμένη πιθανότητα H για το να είναι ένα μήνυμα spam υπολογίζεται ως εξής:

$$H = C^{-1}\left(-2\ln\prod_{w} f(w), 2n\right)$$

Όπου C^{-1} είναι η αντίστροφη chi-square συνάρτηση.

Τέλος υπολογίζεται το

$$I = \frac{1 + H - S}{2}$$

όπου S είναι η αντίστοιχη πιθανότητα του H αλλά για (1-f(w))

Το I είναι ένας δείκτης που παίρνει τιμές κοντά στο 1 όταν ένα μήνυμα είναι spam ενώ παίρνει τιμές κοντά στο 0 όταν ένα μήνυμα είναι ham. Όταν ο δείκτης I πάρει τιμές κοντά στο 0,5 σημαίνει ότι ο αλγόριθμος δεν μπορεί να διακρίνει αν το μήνυμα είναι επιθυμητό ή όχι.

2.5 Sparse Binary Polynomial Hashing

Στο κείμενο αυτό[17] εξετάζεται η τεχνική Sparce Binary Polynomial Hash (SBPH) για τον διαχωρισμό των μηνυμάτων ηλεκτρονικού ταχυδρομείου. Η μέθοδος αυτή αποτελεί γενίκευση της απλής Bayesian μεθόδου αφού υπολογίζει πιθανότητες για ολόκληρες φράσεις και όχι μόνο για λέξεις. Τα βασικά πλεονεκτήματα της μεθόδου σε σχέση με την απλή Bayes είναι:

- Ο αλγόριθμος δίνει μεγαλύτερη ακρίβεια στον διαχωρισμό των μηνυμάτων σε spam και ham από ότι ένας απλός Bayes όπου σύμβολα είναι οι λέξεις.
- Χρειάζεται εκπαίδευση με αρκετά μικρότερο αριθμό μηνυμάτων από ότι ο απλός αλγόριθμος, ώστε να επιτύχει ικανοποιητικό διαχωρισμό των μηνυμάτων.

3.Μεθοδολογία Πειραμάτων

3.1 Υπολογιστικό σύστημα όπου έγιναν τα πειράματα

Για την εκτέλεση των πειραμάτων χρησιμοποιήθηκε ένας υπολογιστής Pentium 3 500Mhz με λειτουργικό σύστημα Gentoo Linux. Δεν χρησιμοποιήθηκαν τα Windows της Microsoft καθώς η πλειοψηφία του λογισμικού διαχωρισμού ανεπιθύμητων μηνυμάτων είτε υπάρχει μόνο σε περιβάλλον Unix είτε είναι σχεδιασμένη να λειτουργεί καλύτερα στο περιβάλλον αυτό. Τα πακέτα εγκαταστάθηκαν με της προεπιλεγμένες ρυθμίσεις τους και δεν έγινε κάποια παραμετροποιημένη μεταγλώτισσή τους.

3.1 Corpus

Χρησιμοποιηθήκαν 3 corpora για την μέτρηση των επιδόσεων των προγραμμάτων. Δύο είναι ελεύθερα διαθέσιμα και ευρέως χρησιμοποιημένα corpus που δίνονται από τους δημιουργούς του spamassassin στην διεύθυνση http://spamassassin.apache.org/publiccorpus/ και ένα corpus με δεδομένα από τα email που δέχθηκε ο γράφων κατά το διάστημα 1/1/2004 – 1/2/2004.

Αναλυτικότερα τα corpus έχουν ως εξής:

1. Ως ham χρησιμοποιήθηκε το 20030228_easy_ham και ως spam το 20030228_spam_2 από τα corpus του spamassassin. Το πρώτο αποτελείται από 2500 μηνύματα ham ενώ το δεύτερο από 1400 spam μηνύματα.
2. Ως ham χρησιμοποιήθηκε το 20030228_easy_ham_2 και ως spam το 20030228_spam από τα corpus του spamassassin. Το πρώτο αποτελείται από 1400 ham μηνύματα ενώ το δεύτερο από 500 spam.
3. Το corpus με δεδομένα από τα προσωπικά μηνύματα του γράφοντος. Το corpus αυτό αποτελείται από 2800 spam μηνύματα και 384 ham μηνύματα. Το γεγονός ότι υπάρχει μεγάλη διαφορά ανάμεσα στα επιθυμητά και ανεπιθύμητα μηνύματα του corpus αυτού αντικατοπτρίζει την σημερινή πραγματικότητα αφού οι περισσότεροι χρήστες καθημερινά λαμβάνουν πολύ περισσότερα ανεπιθύμητα από ότι επιθυμητά μηνύματα.

Ποιο η ποια από τα corpus χρησιμοποιήθηκαν για κάθε μέτρηση το αναφέρουμε στην παράγραφο των μετρήσεων.

3.2 Μορφή ενός email

3.2.1 Μορφή ενός Επιθυμητού Μηνύματος

Στην συνέχεια παρατίθεται ένα ενδεικτικό επιθυμητό μήνημα ηλεκτρονικού ταχυδρομείου. Το μήνυμα αυτό επιλέχθηκε τυχαία από το corpus 1 που αναφέρθηκε προηγουμένως.

```
From quinlan@pathname.com  Fri Aug 23 11:33:57 2002
Return-Path: <quinlan@pathname.com>
Delivered-To: zzzz@localhost.netnoteinc.com
Received: from localhost (localhost [127.0.0.1])
      by phobos.labs.netnoteinc.com (Postfix) with ESMTP id D1C5643F99
      for <zzzz@localhost>; Fri, 23 Aug 2002 06:33:56 -0400 (EDT)
Received: from phobos [127.0.0.1]
      by localhost with IMAP (fetchmail-5.9.0)
      for zzzz@localhost (single-drop); Fri, 23 Aug 2002 11:33:56 +0100 (IST)
Received: from proton.pathname.com
   (adsl-216-103-211-240.dsl.snfc21.pacbell.net [216.103.211.240]) by
   dogma.slashnull.org (8.11.6/8.11.6) with ESMTP id g7NAVFZ20272 for
   <zzzz@spamassassin.taint.org>; Fri, 23 Aug 2002 11:31:15 +0100
Received: from quinlan by proton.pathname.com with local (Exim 3.35 #1
   (Debian)) id 17iBiq-0005K9-00; Fri, 23 Aug 2002 03:31:20 -0700
To: zzzz@spamassassin.taint.org, craig@deersoft.com
Subject: FYI - gone this weekend
Cc: quinlan@pathname.com
Message-Id: <E17iBiq-0005K9-00@proton.pathname.com>
From: Daniel Quinlan <quinlan@pathname.com>
Date: Fri, 23 Aug 2002 03:31:20 -0700


I won't be reading email until Sunday night or so.  Good luck with
```

2.40 and don't do anything I wouldn't do. ;-)

- Dan

Τα μηνύματα ηλεκτρονικού ταχυδρομείου (email) αποτελούνται από δυο τμήματα. Το πρώτο τμήμα είναι αυτό που περιέχει τις κεφαλίδες (headers) του μηνύματος. Οι κεφαλίδες έχουν την μορφή:
<Όνομα κεφαλίδας> : <Περιεχόμενο Κεφαλίδας>
και περιέχουν διάφορες πληροφορίες για το μήνυμα. Ας εξετάσουμε στην συνέχεια μερικές από τις πιο σημαντικές κεφαλίδες των μηνυμάτων ηλεκτρονικού ταχυδρομείου:

- <received>: Η κεφαλίδα αυτή περιέχει πληροφορίες για την διαδρομή που έχει ακολουθήσει ένα μήνυμα από την στιγμή που έφυγε από τον αποστολέα μέχρι να φτάσει στον παραλήπτη. Κάθε σύστημα από το οποίο περνάει προσθέτει και μια κεφαλίδα received.
- <From>: Η κεφαλίδα αυτή περιέχει την ηλεκτρονική διεύθυνση του αποστολέα. Δυστυχώς όμως η κεφαλίδα αυτή μπορεί να πάρει οποιαδήποτε τιμή θέλει ο αποστολέας και δεν είναι απαραίτητα αληθής όπως θα δούμε και στο δείγμα ανεπιθύμητου μηνύματος που ακολουθεί.
- <To>: Η κεφαλίδα αυτή περιέχει την ηλεκτρονική διεύθυνση του παραλήπτη. Δυστυχώς και σε αυτήν την περίπτωση ο αποστολέας μπορεί να βάλει μια ψεύτικη τιμή στην κεφαλίδα αυτή. Όπως φαίνεται από τις δυο κεφαλίδες <from> και <to> είναι αδύνατο βλέποντας μόνο το περιεχόμενο ενός μηνύματος ηλεκτρονικού ταχυδρομείου να βρούμε τον πραγματικό αποστολέα αλλά και τον παραλήπτη του μηνύματος. Ο παραλήπτης του μηνύματος εμφανίζεται ως πληροφορία στον εξυπηρετητή αλληλογραφίας που αναλαμβάνει την παράδοση του μηνύματος, ο πραγματικός αποστολέας όμως μπορεί να μην εμφανίζεται πουθενά. Το γεγονός αυτό εκμεταλλεύονται συνήθως οι αποστολείς ανεπιθύμητων μηνυμάτων αλλάζοντας τις διευθύνσεις του αποστολέα και παραλήπτη που φαίνονται στο μήνυμα σε σχέση με τις πραγματικές.
- <Subject>: Είναι το θέμα του μηνύματος ηλεκτρονικού ταχυδρομείου

- Σε ένα μήνυμα μπορεί να προστεθούν εκτός από τις βασικές κεφαλίδες που αναφέραμε και οποιεσδήποτε άλλες θέλει, είτε ο αποστολέας, είτε οι ενδιάμεσοι εξυπηρετητές και προγράμματα που αναλαμβάνουν την μεταφορά και επεξεργασία του μηνύματος. Τα προγράμματα διαχωρισμού που θα εξετάσουμε προσθέτουν συνήθως και κάποιες δικές τους κεφαλίδες με πληροφορίες όταν εξετάζουν ένα μήνυμα ηλεκτρονικού ταχυδρομείου.

Το δεύτερο τμήμα του μηνύματος είναι το σώμα (body) του μηνύματος που περιέχει την πληροφορία του μηνύματος και είναι αυτό που διαβάζει ο παραλήπτης.

Όπως βλέπουμε το σώμα του μηνύματος μπορεί να είναι πολύ μικρό σε μέγεθος οπότε κανένα από τα προγράμματα που εξετάζουμε δεν βασίζεται μόνο σ' αυτό για την εκπαίδευση των αλγορίθμων του. Επιπλέον οι κεφαλίδες εμφανίζουν πολύ χρήσιμες πληροφορίες που μπορούν τις περισσότερες φορές να διακρίνουν αν ένα μήνυμα είναι επιθυμητό ή όχι χωρίς να ληφθεί υπ' όψη το περιεχόμενο του μηνύματος. Μια τέτοια πληροφορία είναι η κεφαλίδα <from>. Πράγματι αν κάποιο μήνυμα προέρχεται για παράδειγμα από κάποιο συγγενικό πρόσωπο του παραλήπτη είναι μάλλον αδύνατο το μήνυμα αυτό να είναι ανεπιθύμητο άσχετα από το ίδιο το περιεχόμενο του μηνύματος.

3.2.2 Μορφή ενός Ανεπιθύμητου Μηνύματος

```
Return-Path: vafyjerryoil2002@yahoo.co.uk
Delivery-Date: Mon May 27 14:55:16 2002
Received: from 211.185.200.2 ([211.185.200.2]) by dogma.slashnull.org
    (8.11.6/8.11.6) with SMTP id g4RDtCe14573 for <fma@jmason.org>;
    Mon, 27 May 2002 14:55:13 +0100
Message-Id: <200205271355.g4RDtCe14573@dogma.slashnull.org>
From: qtcyJames Dearborn <vafyjerryoil2002@yahoo.co.uk>
To: fma@spamassassin.taint.org
Subject: ADV Oil and Gas Investment tgym
Sender: qtcyJames Dearborn <vafyjerryoil2002@yahoo.co.uk>
MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
```

Date: Mon, 27 May 2002 09:55:24 -0400
X-Keywords:

How would you like a 100% tax free Investment in Oil and Gas wells?
Make over 100% annually and receive monthly tax free Income with
very low risk. If you are liquid for a $10,000 investment, email your
name, address, and phone number to
oilandgaspackage@aol.com and we will send you the information

==========
====
To Unsubscribe from these special mailings:
Forward this mail with "UNSUBSCRIBE" in the subject line to oilandgasremoval@aol.com
==========
====

fghludnbnobfmohbvmsojkdkkf

3.3 Σετ Πακέτων (S/W που χρησιμοποιήθηκε)

3.3.1 Spamassassin

Το Spamassassin[7] είναι ίσως το δημοφιλέστερο από τα προγράμματα διαχωρισμού ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου ενώ είναι και μάλλον το πρώτο πρόγραμμα που έκανε την εμφάνισή του για την καταπολέμηση του φαινόμενου του spam, το οποίο και έγινε ευρέως γνωστό. Θα μπορούσαμε να πούμε ότι το Spamassassin είναι και το πιο πλήρες από τα προγράμματα που εξετάζουμε όσον αφορά τις δυνατότητες του. Πράγματι ο διαχωρισμός με χρήση της μεθόδου Bayes όπως εμφανίστηκε πρώτα στο "A plan for spam"[6] του Paul Graham είναι μόνο μία από τις πολλές μεθόδους που μπορεί να χρησιμοποιήσει το spamassassin για να επιτύχει τον διαχωρισμό των μηνυμάτων ηλεκτρονικού ταχυδρομείου. Τα υπόλοιπα από τα προγράμματα που εξετάσθηκαν βασίζονται μόνο στην μέθοδο

Bayes για να επιτύχουν τον διαχωρισμό. Εδώ να σημειώσουμε ότι στην τρέχουσα εργασία έγινε χρήση μόνο της δυνατότητας διαχωρισμού με Bayes του spamassassin αφού εκεί εστιάζεται και το θέμα της εργασίας.

Μερικά από τα χαρακτηριστικά του spamassassin είναι:

- Χρησιμοποιεί πολλές διαφορετικές μεθόδους για να επιτύχει τον διαχωρισμό των μηνυμάτων
- Διαθέτει μεγάλες δυνατότητες παραμετροποίησης, επιτρέποντας του να ρυθμιστεί ώστε να ταιριάζει στις απαιτήσεις των περισσοτέρων εγκαταστάσεων, ακόμη και μεγάλων εξυπηρετητών που διαχειρίζονται χιλιάδες μηνύματα την μέρα.
- Υπάρχει δυνατότητα ομαδοποίησης των επιλογών των χρηστών
- Ακολουθεί επεκτάσιμη αρχιτεκτονική. Υπάρχει δυνατότητα ανάπτυξης plug-ins που συμπληρώνουν και προσαρμόζουν τις δυνατότητες του βασικού πακέτου.
- Υπάρχει μια μεγάλη και ενεργός κοινότητα χρηστών για το πρόγραμμα
- Χρήση blacklists. Το spamassassin επιτρέπει την χρήση λιστών με μη αποδεκτούς αποστολείς email καθώς και με μη αποδεκτές διευθύνσεις IP αποστολέων.
- Ο χρήστης μπορεί να ρυθμίσει την ευαισθησία του προγράμματος για την απόρριψη των ανεπιθύμητων μηνυμάτων.
- Το spamassassin μπορεί και εκτελείται εξίσου καλά είτε πρόκειται για ένα μικρό σύστημα που ελέγχει μερικές δεκάδες μηνύματα την μέρα είτε για ένα πολυεπεξεργαστικό σύστημα που διαχειρίζεται τα μηνύματα ηλεκτρονικού ταχυδρομείου εκατομμυρίων χρηστών.

Το spamassassin είναι εύκολο να εγκατασταθεί και να παραμετροποιηθεί σε όλες σχεδόν τις σημερινές υπολογιστικές πλατφόρμες, όπως Linux και Windows αν και παραδοσιακά αποδίδει καλύτερα σε περιβάλλον Unix. Το λογισμικό είναι γραμμένο σε γλώσσα perl η οποία και θα πρέπει να υπάρχει εγκατεστημένη στο σύστημα που εκτελείται το spamassassin. Η perl είναι προεγκατεστημένη σε όλες σχεδόν τις διανομές Linux και Unix ενώ διατίθεται δωρεάν και για Windows. Το spamassassin μπορεί να εγκατασταθεί ώστε να επεξεργάζεται τα μηνύματα όλων των χρηστών του

συστήματος αλλά μπορεί και να χρησιμοποιείται μόνο επιλεκτικά στους χρήστες που έχουν πρόβλημα με το spam.

3.3.3 dspam

Το dspam[9] είναι ίσως το πιο σύγχρονο από τα προγράμματα που εξετάσθηκαν στην εργασία αυτή. Είναι αρκετά εξειδικευμένο και δεν συνίσταται η χρήση του σε μεγάλες εγκαταστάσεις παραγωγής. Μερικά από τα χαρακτηριστικά του είναι:

- Βασίζεται σε μεγάλο βαθμό στην εφαρμογή νέων ερευνητικών μεθόδων για την αντιμετώπιση των ανεπιθύμητων μηνυμάτων όπως οι Concept Identification, Message Innoculation, Noise Reduction.
- Μικρές απαιτήσεις επεξεργαστικής ισχύος αφού το πρόγραμμα είναι γραμμένο σε γλώσσα C. Εκτελείται πολύ πιο γρήγορα από ότι προγράμματα γραμμένα σε perl ή python.
- Ευκολία διαχείρισης. Το dspam απαιτεί σε γενικές γραμμές μικρή προσπάθεια και λίγο χρόνο διαχείρισης. Οι διαχειριστές εκτός από την αρχική εγκατάσταση δεν χρειάζεται να ασχολούνται συνεχώς μαζί του.
- Ευκολία χρήσης. Οι απλοί χρήστες χρειάζεται να κάνουν απλώς forward τα false positives και τα false negatives σε μια συγκεκριμένη διεύθυνση ώστε να εκπαιδευτεί σωστά ο αλγόριθμος.
- Υποστήριξη ποικιλίας μηχανών αποθήκευσης της εκπαιδευμένης πληροφορίας, όπως SQLite, Berkeley DB3, Berkeley DB4, MySQL, PostgreSQL και Oracle.
- Υποστήριξη των γνωστότερων Mail Transfer Agents όπως Sendmail, Postfix, Qmail, Courier, και Exim. Μπορεί να εκτελεστεί και ως SMTP Gateway σε περίπτωση που θα πρέπει να συνεργαστεί με κάποιον MTA που δεν ανήκει στην προηγούμενη λίστα.

3.3.4 Spambayes

Το spambayes[8] είναι ένα ακόμη πρόγραμμα διαχωρισμού ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου. Η κυριότερη διαφορά του spambayes με τα υπόλοιπα προγράμματα που εξετάζονται στην εργασία είναι ότι είναι γραμμένο στην γλώσσα python μια σύγχρονη και ευέλικτη γλώσσα προγραμματισμού που επιτρέπει πολύ

γρήγορη ανάπτυξη προγραμμάτων. Οι δημιουργοί του spambayes αναφέρουν ότι το πρόγραμμα αρχικά ξεκίνησε ως μια εφαρμογή της απλής μεθόδου του Paul Graham στο "A plan for Spam"[6]. Σύντομα όμως κύριο μέλημα τους έγινε η χρήση μερικών προχωρημένων τεχνικών που βελτιώνουν τα αποτελέσματα του αλγορίθμου, όπως η "Chi square combining"[15]

Το spambayes όπως αναφέρθηκε είναι γραμμένο σε γλώσσα python οπότε μπορεί να εκτελεστεί σε όλες σχεδόν τις πλατφόρμες όπως Linux, Windows, MacOS X, αρκεί να εγκατασταθεί πρώτα η πλατφόρμα python στο σύστημα.

3.3.5 Spamprobe

Το spamprobe[10] είναι άλλο ένα πρόγραμμα βασισμένο στο "A plan for Spam"[6] του paul graham. Το πρόγραμμα είναι γραμμένο σε C++ και υπόσχεται αρκετά καλές επιδόσεις. Μερικά από τα χαρακτηριστικά του είναι:

- Αυτόματη εκπαίδευση από τα εισερχόμενα μηνύματα ηλεκτρονικού ταχυδρομείου. Ο κάθε χρήστης μπορεί να επέμβει στην διαδικασία της αυτόματης μάθησης
- Χρησιμοποιεί περιβάλλον command line για την εκτέλεσή του και συνεργάζεται με το procmail για την παράδοση και τον διαχωρισμό των μηνυμάτων.
- Χρησιμοποιεί την βιβλιοθήκη βάσης δεδομένων Berkeley DB για γρήγορη εκκίνηση και εκτέλεση του προγράμματος
- Ανιχνεύει και αποκωδικοποιεί συννημένα MIME μηνυμάτων κωδικοποιημένα σε base 64 και quoted printable.
- Λαμβάνει υπ' όψη και φράσεις δυο λέξεων, εκτός από τις απλές λέξεις, για την επίτευξη μεγαλύτερης ακρίβειας στον διαχωρισμό των μηνυμάτων.
- Αγνοεί τα html tags των μηνυμάτων τα οποία συνήθως χρησιμοποιούνται από τους αποστολείς των ανεπιθύμητων μηνυμάτων για να μπερδέψουν τα προγράμματα διαχωρισμού.

Το spamprobe μπορεί να εκτελεστεί σε όλα τα περιβάλλοντα UNIX όπως Linux, Solaris, Freebsd κτλ. Δυστυχώς δεν υπάρχει υποστήριξη για windows αυτήν την στιγμή.

3.3.6 Bogofilter

Το τελευταίο από τα προγράμματα που εξετάζουμε είναι το Bogofilter[11]. Το bogofilter είναι ίσως και το λιγότερο εξελιγμένο από τα προγράμματα διαχωρισμού ανεπιθύμητων μηνυμάτων αφού βασίζεται στην μέθοδο "A plan for Spam"[6] του Paul Graham χωρίς να έχει υλοποιηθεί κάποια πιο εξελιγμένη μέθοδος. Η μέτρηση της επίδοσής του παρουσιάζει ενδιαφέρον για να ελέγξουμε την απόδοση του αλγορίθμου Bayes στην πιο απλή του μορφή. Το bogofilter είναι γραμμένο σε γλώσσα C οπότε εκτελείται και αρκετά γρήγορα. Για την λειτουργία του συνεργάζεται με κάποιο πρόγραμμα όπως το procmail για να παραδώσει τα μηνύματα στους φακέλους των χρηστών. Όπως και τα υπόλοιπα προγράμματα που εξετάσαμε τα οποία είναι γραμμένα σε γλώσσα C, έτσι και το bogofilter εκτελείται αποκλειστικά σε περιβάλλον linux χωρίς να υπάρχει κάποια έκδοση για Windows.

3.4 Μετρήσεις

Οι μετρήσεις που πραγματοποιήθηκαν είχαν ως σκοπό να περιγράψουν με τον καλύτερο δυνατό τρόπο την λειτουργία των προγραμμάτων κάτω από διάφορες συνθήκες λειτουργίας που είναι δυνατό να εμφανιστούν κατά την χρήση των προγραμμάτων σε πραγματικό περιβάλλον χρήσης. Διακρίναμε τις ακόλουθες τρεις περιπτώσεις τις οποίες και εξετάσαμε:

1. Εκπαίδευση με προσωπικά δεδομένα του χρήστη
2. Εκπαίδευση με ένα σύνολο γενικών μηνυμάτων και μετά χρήση για τον διαχωρισμό των προσωπικών μηνυμάτων ενός χρήστη
3. Μεταβολή της απόδοσης των προγραμμάτων ανάλογα με το πλήθος των μηνυμάτων που χρησιμοποιήθηκε για την εκπαίδευση

στην συνέχεια θα αναφερθούμε αναλυτικότερα σε κάθε μια από τις μετρήσεις.

Η αξιολόγηση που πραγματοποιήθηκε αφορά μόνο την απόδοση των αλγορίθμων διαχωρισμού των μηνυμάτων που περιλαμβάνουν τα προγράμματα και όχι την συνολική εικόνα, τις λειτουργίες και τις δυνατότητες διαχείρισης των προγραμμάτων αυτών. Επίσης δεν αξιολογήθηκε η ταχύτητα εκτέλεσης των προγραμμάτων, παρά το γεγονός ότι αποτελεί έναν σημαντικό παράγοντα στην επιλογή ενός λογισμικού για κεντρική εγκατάσταση σε εξυπηρετητή πολλών χρηστών. Αυτό έγινε αφού δεν

υπάρχει ουσιαστικά αντικειμενικός τρόπος αξιολόγησης της ταχύτητας, η οποία σε μεγάλο βαθμό εξαρτάται συνήθως από την διαμόρφωση του συστήματος στο οποίο πραγματοποιούνται οι δοκιμές.

Δεδομένου του πλήθους των προγραμμάτων που εξετάστηκαν, καθώς και της μεγάλης απαίτησης σε χρόνο για την εκτέλεση των μετρήσεων, κρίθηκε σκόπιμη η ανάπτυξη λογισμικού για την αυτοματοποίηση της διαδικασίας των μετρήσεων. Στην απόφαση αυτή συνέβαλε και το γεγονός ότι τα προγράμματα αυτά γράφηκαν για να διαχωρίζουν τα μηνύματα σε πραγματικές συνθήκες χωρίς να υπάρχει τις περισσότερες φορές κάποιος εύκολος τρόπος για την μέτρηση της απόδοσής τους.

Το λογισμικό αναπτύχθηκε με χρήση της γλώσσας perl σε περιβάλλον gentoo linux και αποτελείται από 3 ανεξάρτητα τμήματα:

1. Το λογισμικό εκμάθησης. Το λογισμικό αυτό είναι ανεξάρτητο για κάθε ένα από τα προγράμματα που ελεγχθήκαν. Αναλαμβάνει να εκπαιδεύσει κάθε ένα από τα προγράμματα με το corpus το οποίο έχει επιλεγεί κάθε φορά.
2. Το λογισμικό ελέγχου. Το λογισμικό αυτό είναι επίσης ανεξάρτητο για κάθε ένα από τα προγράμματα. Αναλαμβάνει να ελέγξει την απόδοση καθενός από τα προγράμματα με το corpus που επιλέγεται κάθε φορά και καταγράφει τα αποτελέσματα της μέτρησης σε ένα αρχείο.
3. Το λογισμικό μετασχηματισμού. Το λογισμικό αυτό αναλαμβάνει να μετατρέψει τα αποτελέσματα κάθε προγράμματος στην κοινή μορφή που επιλέξαμε για την παρουσίαση.

Το λογισμικό που αναπτύχθηκε παρατίθεται στο παράρτημα Α.

3.4.1 Μέτρηση 1

Στην πρώτη μέτρηση εξετάζουμε την λειτουργία των προγραμμάτων υπό φυσιολογικές συνθήκες. Λέγοντας «φυσιολογικές συνθήκες» εννοούμε ότι τα προγράμματα έχουν ήδη εκπαιδευτεί με ικανό αριθμό επιθυμητών και ανεπιθύμητων μηνυμάτων ώστε να μπορούν να αποδώσουν την βέλτιστη απόδοσή τους. Για την μέτρηση αυτή πραγματοποιήθηκε εκπαίδευση με το corpus 1 και έλεγχος της απόδοσης με το corpus 2.

3.4.2 Μέτρηση 2

Στην μέτρηση 2 εξετάζουμε την λειτουργία των προγραμμάτων κάτω από ένα άλλο πιθανό σενάριο. Στην περίπτωση αυτή γίνεται πάλι εκπαίδευση με το corpus 1 αλλά έλεγχος της απόδοσης με το corpus 3. Μια πιθανή ρεαλιστική εφαρμογή του σεναρίου αυτού αφορά την περίπτωση που κάποιος χρήστης ή κάποιος οργανισμός, που αποφασίζει να χρησιμοποιήσει κάποιο από τα προγράμματα αυτά, δεν εκπαιδεύει με προσωπικά δεδομένα τον αλγόριθμο αλλά χρησιμοποιεί για την εκπαίδευση κάποιο έτοιμο σύνολο που βρίσκει στο διαδίκτυο. Η μέτρηση 2 εξετάζει αυτήν ακριβώς την περίπτωση αφού εκπαίδευση γίνεται με ένα γενικό corpus από το site του spamassassin ενώ έλεγχος γίνεται με το corpus που δημιουργήθηκε με τα προσωπικά μηνύματα του γράφοντος.

3.4.3 Μέτρηση 3

Στην μέτρηση 3 εξετάζουμε την μεταβολή της απόδοσης των προγραμμάτων ανάλογα με το μέγεθος του corpus εκπαίδευσης. Η μέτρηση αυτή είναι πολύ σημαντική αφού δίνει μια καθαρή εικόνα της επίδρασης που έχει ο όγκος των δεδομένων που χρησιμοποιούμε για εκπαίδευση στην απόδοση των προγραμμάτων. Πολλές φορές δεν είναι εφικτό, ο χρήστης ενός τέτοιου προγράμματος, είτε πρόκειται για κάποιον μεμονωμένο χρήστη είτε για τον διαχειριστή ενός συστήματος που εξυπηρετεί πολλούς χρήστες, να διαθέτει ένα μεγάλο πλήθος κατάλληλων μηνυμάτων ώστε να εκπαιδεύσει πλήρως τους αλγόριθμους των προγραμμάτων αυτών. Ιδανικά θα θέλαμε τα προγράμματα να λειτουργούν σωστά ακόμη και με ελάχιστο αριθμό μηνυμάτων. Αυτό ακριβώς εξετάζει αυτή η μέτρηση. Την απόκριση δηλαδή των αλγορίθμων μεταβάλλοντας το πλήθος των μηνυμάτων εκπαίδευσης. Όπως και στην μέτρηση 1, για την εκπαίδευση χρησιμοποιούμε το corpus 1 ενώ για τον έλεγχο της απόδοσης, το corpus 2. Στην περίπτωση αυτή όμως πραγματοποιούμε τρεις διαφορετικές μετρήσεις για τρία διαφορετικά μεγέθη του corpus 1. Πραγματοποιούμε μια μέτρηση με το 25% των μηνυμάτων του corpus 1, μια μέτρηση με 50% των μηνυμάτων και μια μέτρηση με 100% των μηνυμάτων. Στην συνέχεια εξετάζουμε την μεταβολή της απόδοσης των αλγορίθμων ανάλογα με τις τρεις διαφορετικές τιμές του πλήθους των μηνυμάτων του corpus εκπαίδευσης.

3.5 Μέθοδος Αξιολόγησης

Για την αξιολόγηση της απόδοσης των προγραμμάτων θα χρησιμοποιήσουμε τον υπολογισμό των ακόλουθων πιθανοτήτων[2]:

- Την πιθανότητα κατάταξης ενός μηνύματος ως ανεπιθύμητο ενώ αυτό είναι επιθυμητό. Για ένα τέτοιο μήνυμα χρησιμοποιούμε τον όρο False Positive ενώ την πιθανότητα του την ορίζουμε ως εξής:

  $$P_{FP} = \frac{n_{ham \to spam}}{n_{ham}}$$

  όπου $n_{ham \to spam}$ είναι ο αριθμός των μηνυμάτων που είναι επιθυμητά και κατατάσσονται ως ανεπιθύμητα ενώ n_{ham} είναι ο συνολικός αριθμός των επιθυμητών μηνυμάτων.
- Την πιθανότητα κατάταξης ενός μηνύματος ως επιθυμητό ενώ αυτό είναι ανεπιθύμητο. Για ένα τέτοιο μήνυμα χρησιμοποιούμε τον όρο False Negative ενώ την πιθανότητα του την ορίζουμε ως εξής:

  $$P_{FN} = \frac{n_{spam \to ham}}{n_{spam}}$$

  όπου $n_{spam \to ham}$ είναι ο αριθμός των μηνυμάτων που είναι ανεπιθύμητα και κατατάσσονται ως επιθυμητά, ενώ n_{spam} είναι ο συνολικός αριθμός των ανεπιθύμητων μηνυμάτων.

Οι δυο αυτές πιθανότητες ορίζουν η καθεμιά από ένα σφάλμα. Η πρώτη ορίζει το σφάλμα στην ανίχνευση των επιθυμητών μηνυμάτων και η δεύτερη ορίζει το σφάλμα στην ανίχνευση των ανεπιθύμητων μηνυμάτων. Στην συνέχεια θα υπολογίσουμε και τις δυο αυτές πιθανότητες για τα προγράμματα τα οποία εξετάζουμε στην παρούσα εργασία και θα τις χρησιμοποιήσουμε ως μέτρο της απόδοσης των προγραμμάτων αυτών.

Μέχρι στιγμής στα δυο μέτρα της απόδοσης που αναφέρθηκαν δεν λήφθηκε υπόψη το γεγονός ότι τις περισσότερες φορές κατά την λειτουργία αυτών των προγραμμάτων εμφανίζουν διαφορετικό βάρος οι δυο πιθανότητες που αναφέραμε. Για να γίνει καλύτερα κατανοητό αυτό παραθέτουμε ένα παράδειγμα. Έστω ότι κάποιος χρήστης χρησιμοποιεί ένα πρόγραμμα ώστε να διαχωρίζει τα ανεπιθύμητα μηνύματα σε έναν διαφορετικό φάκελο (π.χ. στο φάκελο spam) ενώ τα υπόλοιπα παραδίδονται κανονικά στο Inbox του. Όποτε ο χρήστης έχει χρόνο ελέγχει τα μηνύματα του φακέλου spam για να δει μήπως υπάρχει κάποιο επιθυμητό μήνυμα αλλά πολλές φορές τα σβήνει χωρίς καν να τα κοιτάξει. Στο παράδειγμα αυτό φαίνεται ότι διαφορετική βαρύτητα έχουν τα δυο σφάλματα που προαναφέραμε. Στην περίπτωση που γίνει λάθος και κάποιο ανεπιθύμητο μήνυμα καταταγεί ως επιθυμητό (False Negative), τότε απλώς εμφανίζεται ένα επιπλέον ανεπιθύμητο στην εισερχόμενη αλληλογραφία του χρήστη με αποτέλεσμα στην χειρότερη περίπτωση το χάσιμο λίγου χρόνου για τον έλεγχό του. Στην περίπτωση όμως της λάθος κατάταξης ενός επιθυμητού μηνύματος ως ανεπιθύμητο (False Positive), τότε οι συνέπειες είναι πολύ σοβαρότερες αφού στην καλύτερη περίπτωση ο χρήστης θα αργήσει να το δει ενώ στην χειρότερη θα το αγνοήσει τελείως διαγράφοντας τα περιεχόμενα του φακέλου με τα ανεπιθύμητα μηνύματα. Για τον λόγο αυτό στα δυο μέτρα, που αναφέρθηκαν προηγουμένως θα προσθέσουμε και την συνδυασμένη πιθανότητα σφάλματος που την ορίζουμε ως:

$$P_{err} = \frac{\lambda \cdot n_{ham \to spam} + n_{spam \to ham}}{\lambda \cdot n_{ham} + n_{spam}}$$

Αν $n_{ham}=n_{spam}$ τότε η συνδυασμένη πιθανότητα σφάλματος παίρνει την μορφή:

$$P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1}$$

Ο συντελεστής λ ορίζει την βαρύτητα του σφάλματος P_{FP} (πιθανότητα εμφάνισης False Positive) σε σχέση με το σφάλμα P_{FN} (πιθανότητα εμφάνισης False Negative).

Στα ακόλουθα αποτελέσματα των μετρήσεων θα παραθέσουμε αποτελέσματα με δυο διαφορετικές τιμές του λ. Μια μέτρηση θα υπολογιστεί με λ=1 που σημαίνει ότι θεωρούμε ότι τα δυο σφάλματα έχουν το ίδιο κόστος, ενώ μια μέτρηση θα υπολογιστεί με λ=9 που σημαίνει ότι η λανθασμένη κατάταξη ενός επιθυμητού

μηνύματος είναι 9 φορές βαρύτερη από την λανθασμένη κατάταξη ενός ανεπιθύμητου μηνύματος.

4.Παρουσίαση και σχολιασμός αποτελεσμάτων

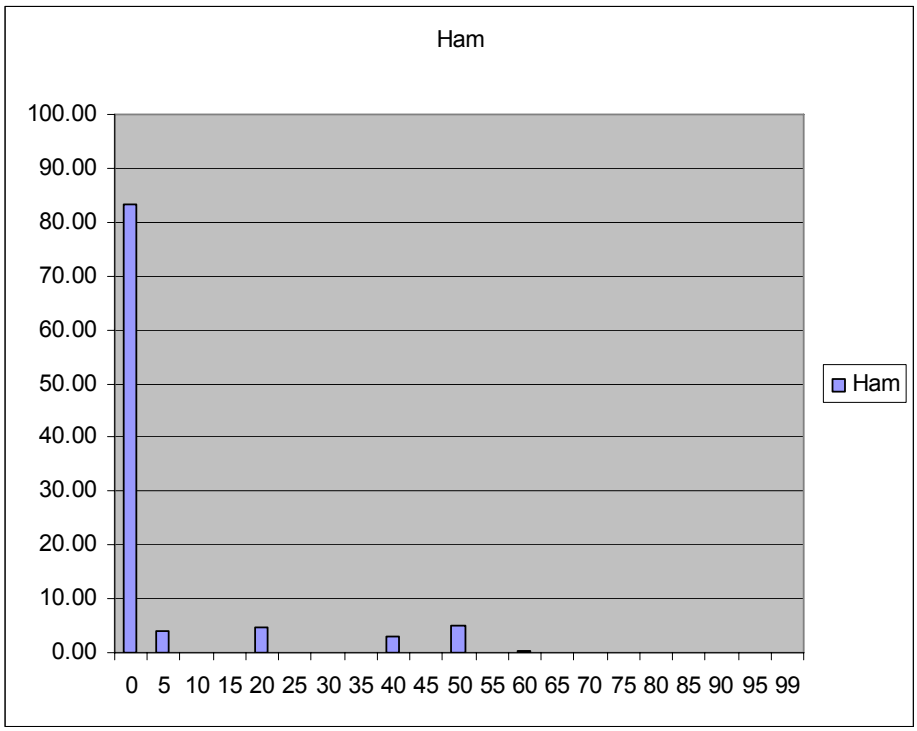
4.1 Συμπεράσματα και σχήματα

4.1.1 Μέτρηση 1

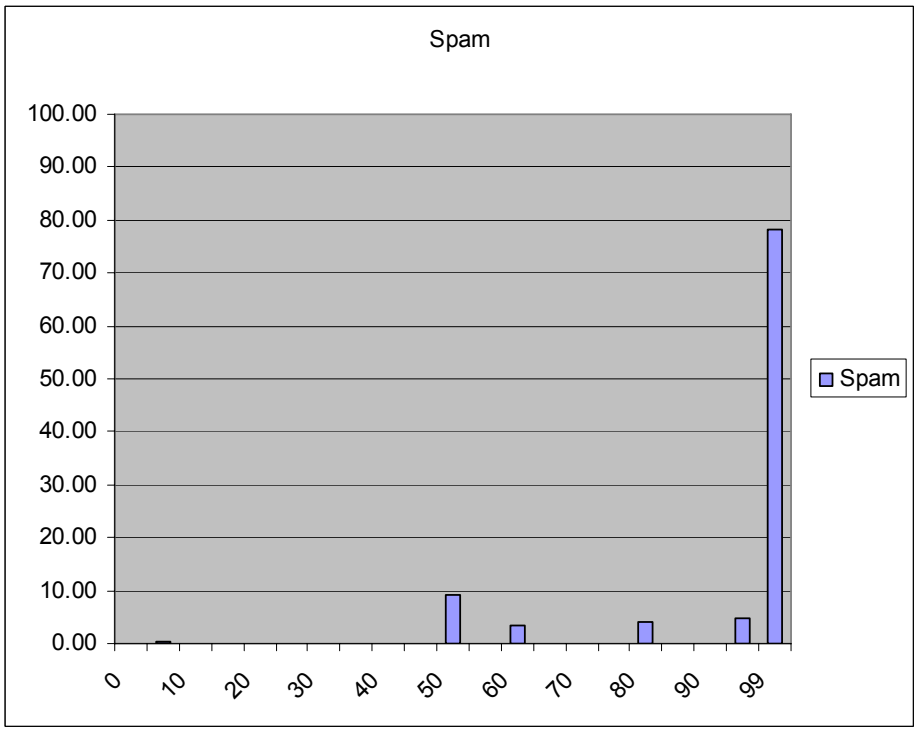
Τα σχήματα της πρώτης μέτρησης εμφανίζουν την ποσοστιαία κατανομή των μηνυμάτων σε βαθμούς «πιθανότητας να είναι ανεπιθύμητα». Έτσι όταν κάποια μηνύματα εμφανίζουν πιθανότητα 0, σημαίνει ότι το λογισμικό είναι βέβαιο ότι τα μηνύματα αυτά είναι επιθυμητά, ενώ τα μηνύματα που εμφανίζουν πιθανότητα 100 σημαίνει ότι το λογισμικό είναι βέβαιο ότι τα μηνύματα είναι ανεπιθύμητα. Για τα μηνύματα που εμφανίζουν πιθανότητα γύρω στο 50 το εκάστοτε λογισμικό αδυνατεί να αποφασίσει αν αυτά είναι επιθυμητά η ανεπιθύμητα. Στα διαγράμματα οι διάφορες πιθανότητες 0-100 εμφανίζονται στον άξονα χ ενώ το ποσοστό μηνυμάτων που αντιστοιχεί στην κάθε πιθανότητα στον άξονα y.

Για τον υπολογισμό των σφαλμάτων θα υποθέσουμε ότι τα μηνύματα με πιθανότητες 0-55% θεωρούνται επιθυμητά ενώ τα μηνύματα με πιθανότητες 55-100% θεωρούνται ανεπιθύμητα.

4.1.1.1 Spamassassin



Εικ. 1 Μέτρηση 1 Spamassassin Ham



Εικ. 2 Μέτρηση 1 Spamassassin Spam

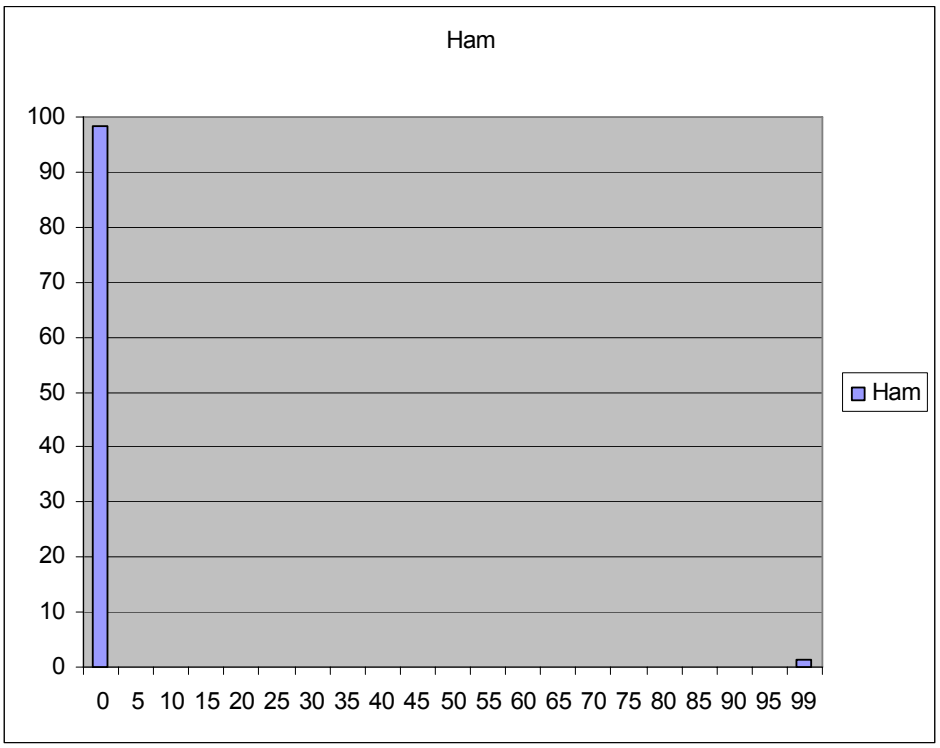
Στην πρώτη μέτρηση το Spamassassin αποδίδει αρκετά καλά στον διαχωρισμό των επιθυμητων μηνυμάτων. Τα περισσότερα κατατάσονται από το πρόγραμμα με πλήρη εμπιστοσύνη ως επιθυμητά (πιθανότητα 0%). Ωστόσο εμφανίζεται και ένα μικρό ποσοστό μηνυμάτων, με πιθανότητα γύρω στο 50%, τα οποία το spamassassin αδυνατεί να κατατάξει σε μια από τις δυο κατηγορίες spam η ham. Η πιθανότητα σφάλματος False Positive είναι P_{FP}=0.0035

Στην περίπτωση διαχωρισμού των ανεπιθύμητων μηνυμάτων το spamassassin αποδίδει αρκετά καλά (λίγο χειρότερα όμως από την περίπτωση διαχωρισμού των επιθυμητών μηνυμάτων. Βλέπουμε και εδώ ένα ποσοστό γύρω στο 10% των μηνυμάτων που το Spamassassin αδυνατεί ουσιαστικά να διαχωρίσει (εμφανίζουν πιθανότητα γύρω στο 50%).

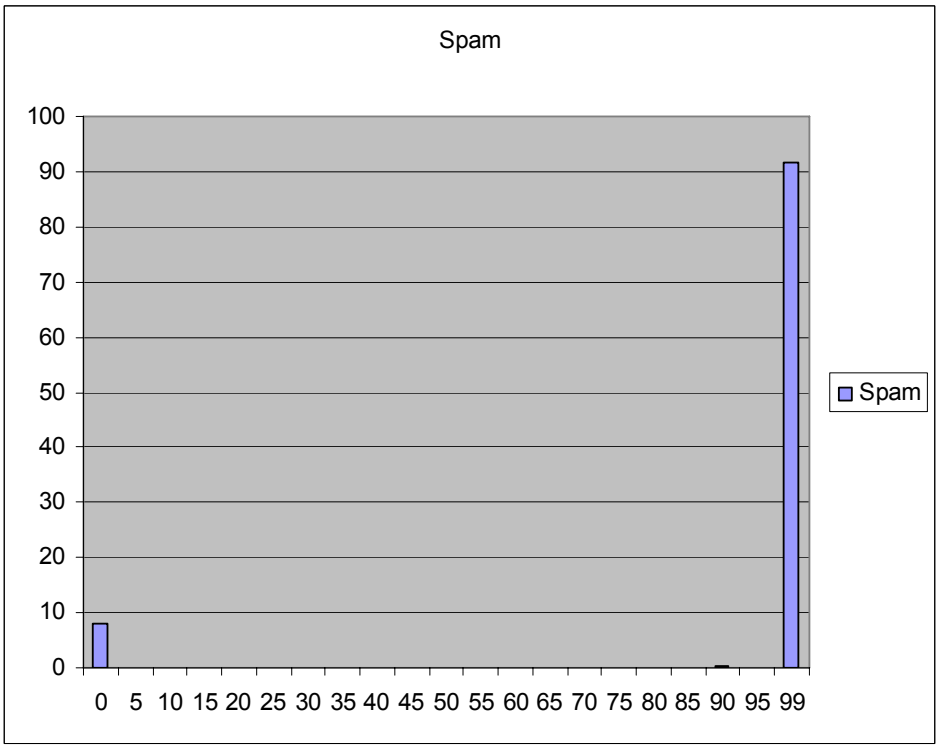
Το σφάλμα False Negative είναι P_{FN}= 0.094. Το σφάλμα αυτό προκύπτει μεγαλύτερο από το αντίστοιχο σφάλμα για τα επιθυμητά μηνύματα γιατί όπως αναφέραμε στην αρχή συμπεριλαμβάνουμε σ' αυτό το σφάλμα και τα ανεπιθύμητα μηνύματα τα οποία το πρόγραμμα δυσκολεύεται να κατατάξει σε μια από τις δυο κατηγορίες. Ο λόγος που επιλέγουμε το 55% ως σημείο διαχωρισμού μεταξύ spam και ham και όχι το 45 ή 50 είναι γιατί όπως αναφέρθηκε προηγουμένως το κόστος λάθος κατάταξης ενός ανεπιθύμητου μηνύματος (False Negative) είναι συνήθως μικρότερο από το κόστος λάθος κατάταξης ενός επιθυμητού μηνύματος (False Positive).
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.05$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = \frac{9 \cdot 0.0035 + 0.094}{10} = 0.01255$

4.1.1.2 Dspam



Εικ. 3 Μέτρηση 1 Dspam Ham



Εικ. 4 Μέτρηση 1 Dspam Spam

Από τα διαγράμματα του Dspam παρατηρούμε ότι υπάρχει μεγάλη πόλωση των αποτελεσμάτων. Πράγματι κατατάσσει την πλειοψηφία των μηνυμάτων είτε ως σίγουρα spam (πιθανότητα 100%) είτε ως σίγουρα ham (πιθανότητα 0%)

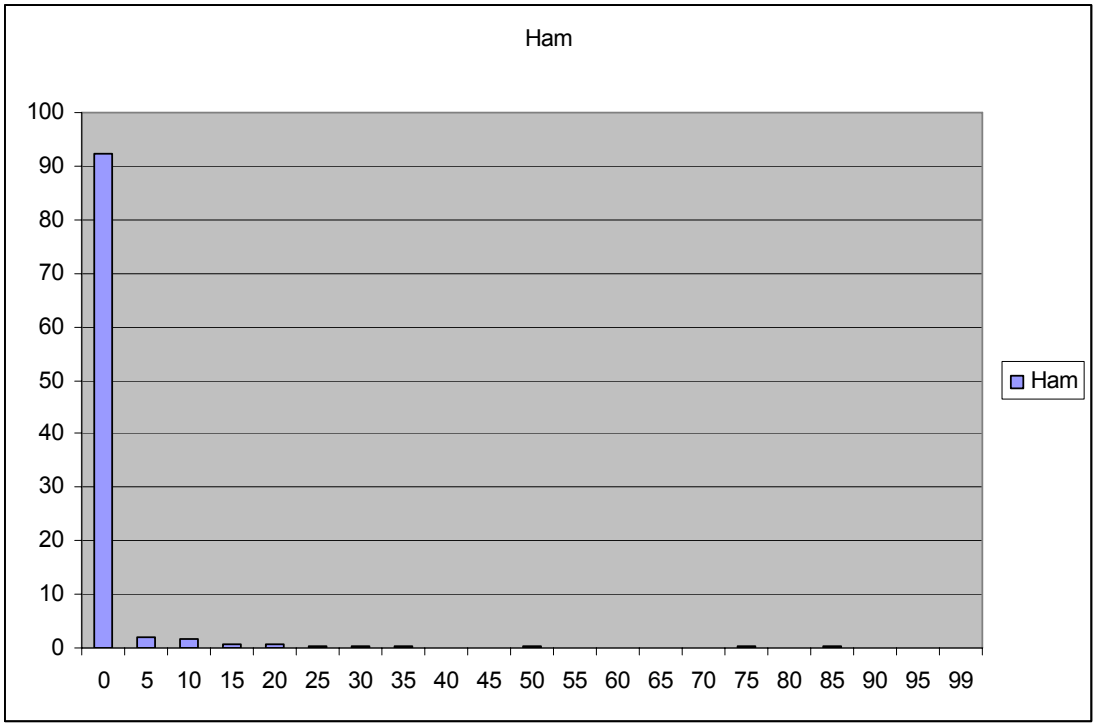
Από το διάγραμμα διαχωρισμού των επιθυμητών μηνυμάτων βλέπουμε ότι το dspam κατατάσσει σωστά ένα πολύ μεγάλο ποσοστό των μηνυμάτων ως επιθυμητά. Το ποσοστό αυτό είναι περίπου 99%. Ωστόσο τα επιθυμητά μηνύματα που αποτυγχάνει να διαχωρίσει σωστά κατατάσσονται ως σίγουρα spam. Αυτό είναι ένα αρκετά σοβαρό μειονέκτημα αφού στις περισσότερες περιπτώσεις πραγματικής χρήσης ενός τέτοιου προγράμματος τα False Positive σφάλματα έχουν πολύ μεγαλύτερο κόστος από τα False Negatives. Στην περίπτωση του dspam όσο και αν μετατοπίσουμε το σημείο διαχωρισμού ham-spam (που για τις μετρήσεις μας το επιλέξαμε στο 55%) δεν θα μπορέσουμε να μειώσουμε το σημαντικό σφάλμα των False Positives που είναι P_{FP}=0.015

Στο διάγραμμα των ανεπιθύμητων μηνυμάτων παρατηρούμε περίπου την ίδια εικόνα με το διάγραμμα των επιθυμητών. Το dspam μπορεί και κατατάσσει σωστά ένα μεγάλο ποσοστό των ανεπιθύμητων μηνυμάτων, ωστόσο ένα σημαντικό ποσοστό κατατάσσεται ως σίγουρο ham (πιθανότητα 0%). Το σφάλμα False Negatives είναι για την περίπτωση του dspam P_{FN}=0.08

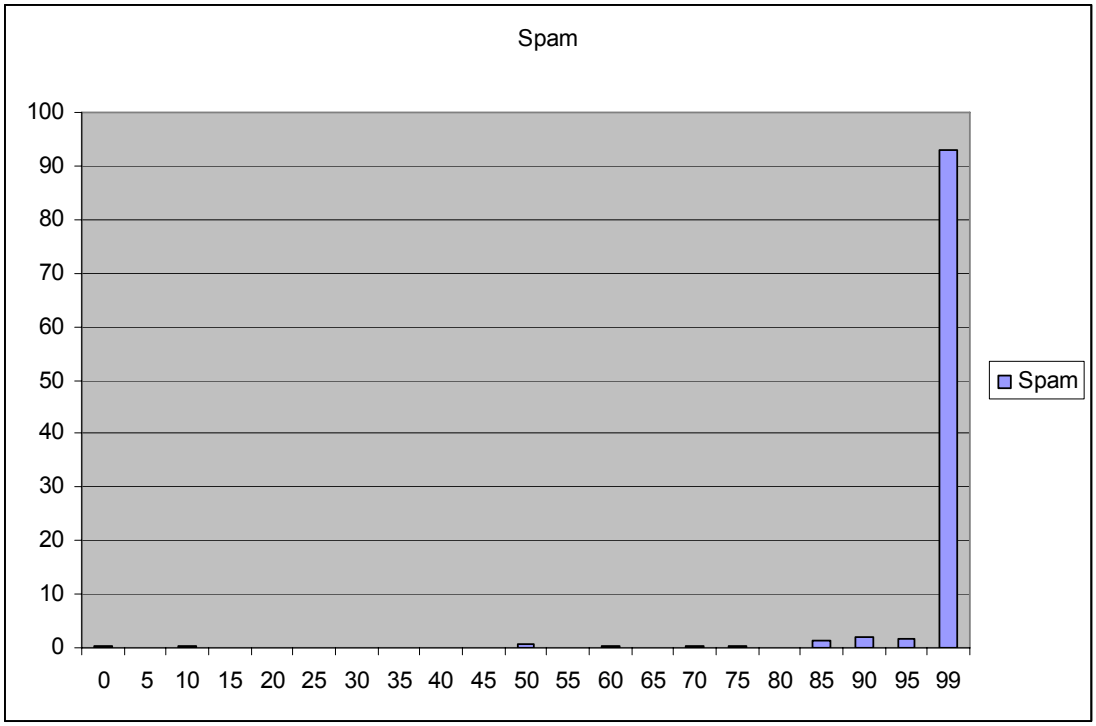
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.05$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.0215$

4.1.1.3 Spambayes



Εικ. 5 Μέτρηση 1 Spambayes Ham



Εικ. 6 Μέτρηση 1 Spambayes Spam

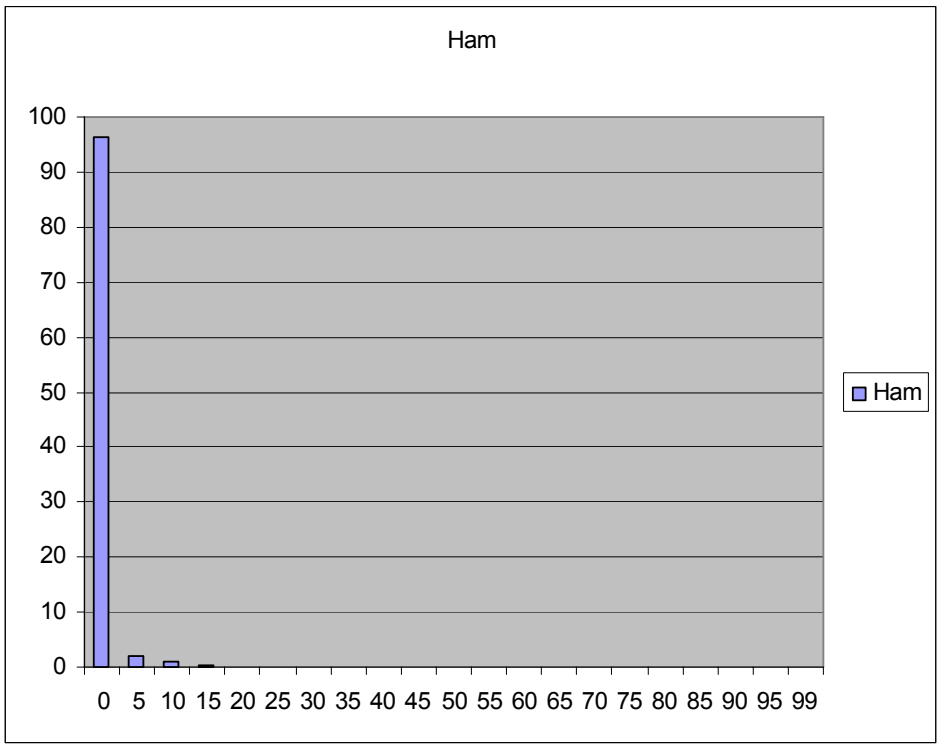
Στην πρώτη μέτρηση το Spambayes αποδίδει αρκετά καλά στον διαχωρισμό των επιθυμητών μηνυμάτων. Τα περισσότερα κατατάσσονται από το πρόγραμμα με πλήρη εμπιστοσύνη ως επιθυμητά (πιθανότητα 0%).Τα υπόλοιπα μηνύματα που το spambayes είτε δυσκολεύται να κατατάξει είτε τα κατατάσσει λάθος μοιράζονται σε διάφορες πιθανότητες χωρίς να παρουσιάζεται ιδιαίτερη συγκέντρωση ούτε στην περιοχή των μηνυμάτων που αδυνατεί να διαχωρίσει (κοντά στο 50%) ούτε στην περιοχή των μηνυμάτων που κατατάσσονται εντελώς εσφαλμένα (κοντά στο 100%). Το σφάλμα False Positive είναι P_{FP}=0.0085

Και στην περίπτωση διαχωρισμού των ανεπιθύμητων μηνυμάτων η εικόνα είναι παρόμοια με τον διαχωρισμό των επιθυμητών μηνυμάτων. Τα περισσότερα αναγνωρίζονται με πλήρη εμπιστοσύνη ως spam από το πρόγραμμα. Τα υπόλοιπα μηνύματα που το spambayes είτε δυσκολεύται να κατατάξει είτε τα κατατάσσει λάθος μοιράζονται σε διάφορες πιθανότητες χωρίς να παρουσιάζεται ιδιαίτερη συγκέντρωση ούτε στην περιοχή των μηνυμάτων που αδυνατεί να διαχωρίσει (κοντά στο 50%) ούτε στην περιοχή των μηνυμάτων που κατατάσσονται εντελώς εσφαλμένα ως επιθυμητά (κοντά στο 0%). Το σφάλμα False Negative είναι P_{FN}=0.01

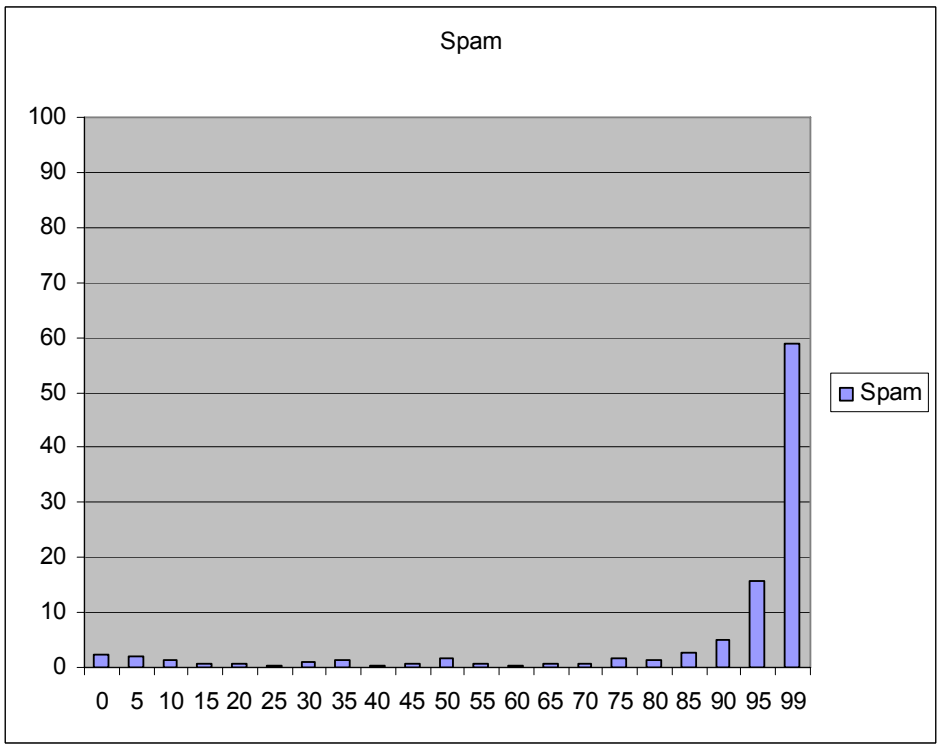
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.009$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.0087$

4.1.1.4 Spamprobe



Εικ. 7 Μέτρηση 1 Spamprobe Ham



Εικ. 8 Μέτρηση 1 Spamprobe Spam

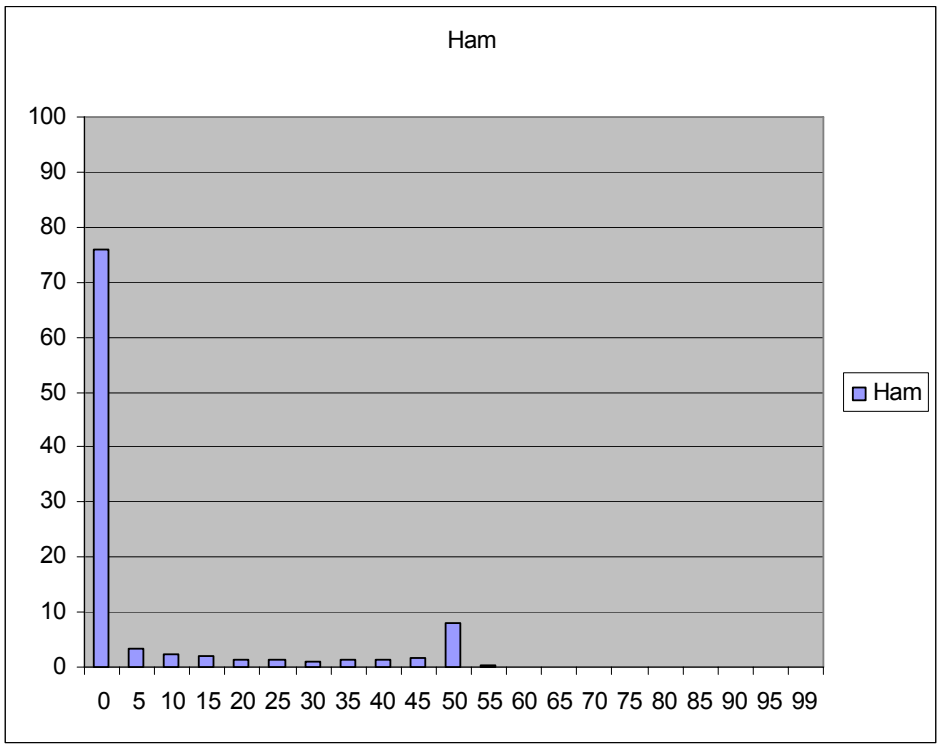
Από το πρώτο διάγραμμα παρατηρούμε μια πολύ καλή απόδοση του spamprobe στην περίπτωση κατάταξης των επιθυμητών μηνυμάτων. Πράγματι η πλειοψηφία κατατάσσεται σωστά ως επιθυμητά με πλήρη εμπιστοσύνη (πιθανότητα 0%) ενώ τα περισσότερα από τα υπόλοιπα κατατάσσονται σωστά με αρκετά μεγάλη εμπιστοσύνη (πιθανότητες 5-15%). Δεν παρατηρούμε συγκέντρωση μηνυμάτων στην περιοχή 50% που αφορά μηνύματα που το πρόγραμμα δυσκολεύεται να κατατάξει, ούτε στην περιοχή 100% που αφορά μηνύματα που το πρόγραμμα κατατάσσει εντελώς λάθος. Έτσι το σφάλμα False Positive για την περίπτωση του Spamprobe παίρνει την πολύ μικρή τιμή P_{FP}=0.0014

Στην περίπτωση κατάταξης των ανεπιθύμητων μηνυμάτων παρατηρούμε μια διαφορετική εικόνα από το spamprobe. Το πρόγραμμα κατατάσσει με πλήρη εμπιστοσύνη ως ανεπιθύμητα ένα μικρό σχετικά ποσοστό (περίπου 60%) των μηνυμάτων. Πολλά από τα υπόλοιπα μηνύματα βρίσκονται διασκορπισμένα σε όλο το φάσμα των πιθανοτήτων είτε με σωστή είτε με λάθος κατάταξη. Σίγουρα η απόδοση του spamprobe δεν κρίνεται ικανοποιητική αν την συγκρίνουμε με τα υπόλοιπα προγράμματα στον τομέα αυτό. Το σφάλμα False Negative είναι P_{FN}=0.126

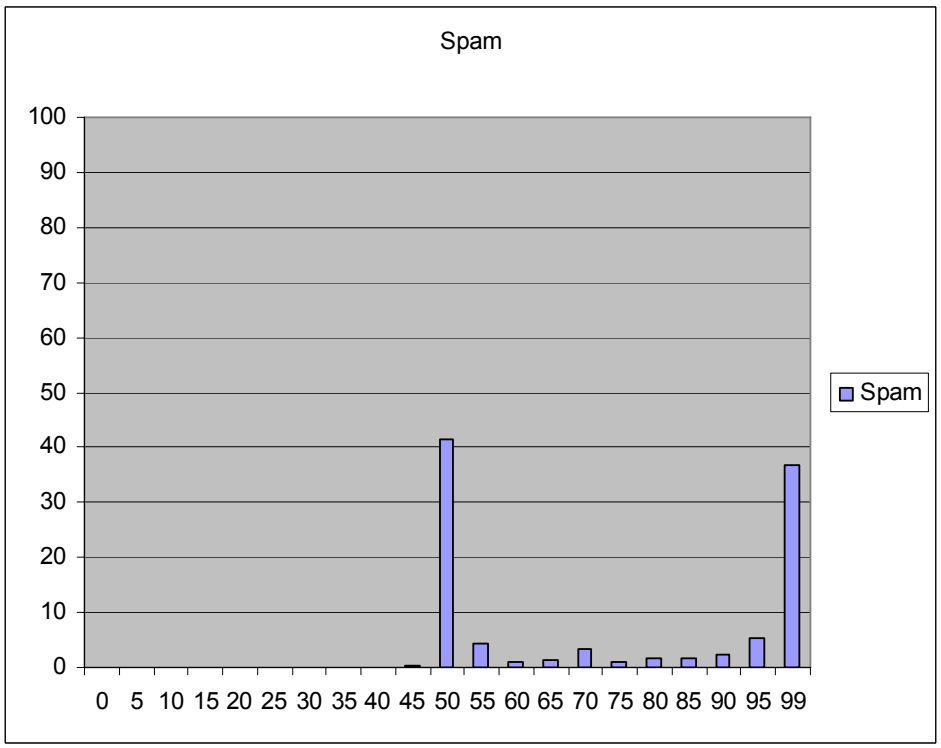
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.064$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.0139$

4.1.1.5 Bogofilter



Εικ. 9 Μέτρηση 1 Bogofilter Ham



Εικ. 10 Μέτρηση 1 Bogofilter Spam

Από το πρώτο διάγραμμα του Bogofilter βλέπουμε ότι το πρόγραμμα κατατάσσει με πλήρη εμπιστοσύνη ένα μέσο αριθμό μηνυμάτων (περίπου το 75%). Πολλά από τα υπόλοιπα μηνύματα, τα οποία έχουν καταταγεί σωστά αλλά με μικρότερη εμπιστοσύνη, βρίσκονται διασκορπισμένα στις πιθανότητες 5-45%. Στην πιθανότητα 50% φαίνεται μια αρκετά μεγάλη συγκέντρωση μηνυμάτων (περίπου το 8%) γεγονός που σημαίνει ότι το bogofilter αδυνατεί να αποφασίσει για τα μηνύματα αυτά. Ωστόσο πολύ μικρός αριθμός μηνυμάτων κατατάσσεται λάθος (πιθανότητες 55-100%). Το σφάλμα False Positive είναι P_{FP}=0.0021

Στην περίπτωση του διαχωρισμού των ανεπιθύμητων μηνυμάτων το bogofilter παρουσιάζει παρόμοια εικόνα με την περίπτωση διαχωρισμού των επιθυμητών μηνυμάτων, μόνο που εδώ η απόδοση είναι χειρότερη. Το πρόγραμμα κατατάσσει ως ανεπιθύμητα μηνύματα με απόλυτη εμπιστοσύνη μόλις το 37% των μηνυμάτων. Για ένα πολύ μεγάλο ποσοστό ανεπιθύμητων μηνημάτων, περίπου 42%, το bogofilter αδυνατεί να τα κατατάξει είτε ως επιθυμητα είτε ως ανεπιθύμητα. Στην περίπτωση αυτή εμφανίζεται ένα πολύ μεγάλο σφάλμα False Negative που είναι P_{FN}=0.418

Το συνολικό σφάλμα είναι:

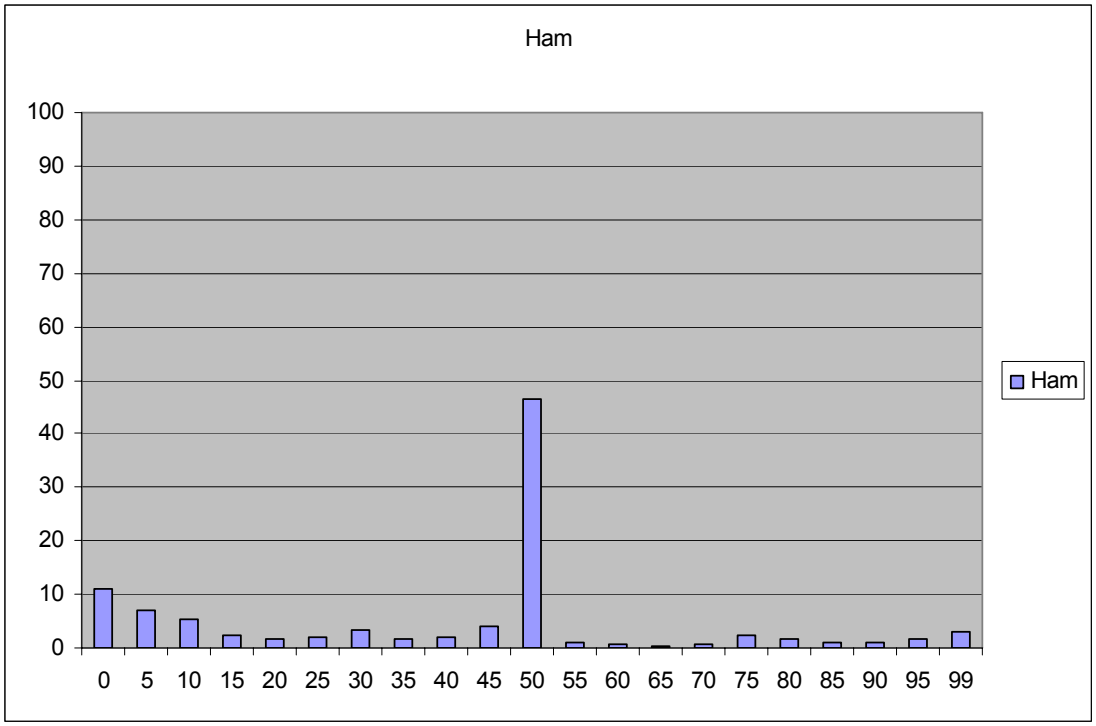
- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.21$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.0437$

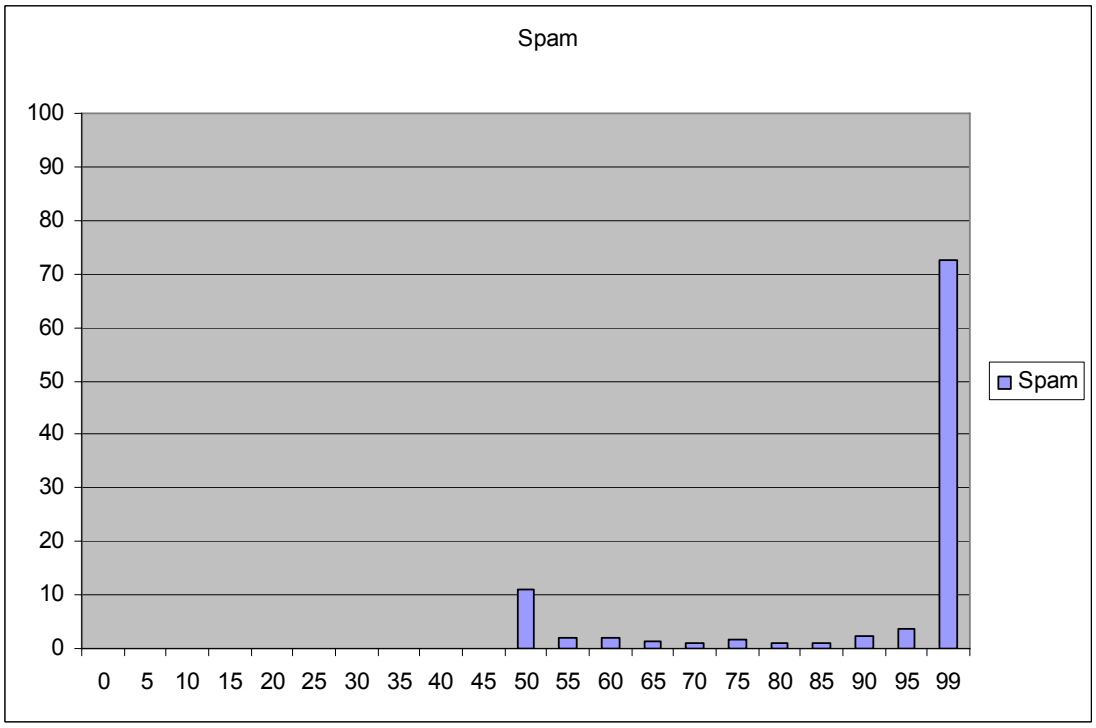
4.1.2 Μέτρηση 2

Τα σχήματα της δεύτερης μέτρησης εμφανίζουν την ποσοστιαία κατανομή των μηνυμάτων σε βαθμούς «πιθανότητας να είναι ανεπιθύμητα». Έτσι όταν κάποια μηνύματα εμφανίζουν πιθανότητα 0, σημαίνει ότι το λογισμικό είναι βέβαιο ότι τα μηνύματα αυτά είναι επιθυμητά, ενώ τα μηνύματα που εμφανίζουν πιθανότητα 100 σημαίνει ότι το λογισμικό είναι βέβαιο ότι τα μηνύματα είναι ανεπιθύμητα. Για τα μηνύματα που εμφανίζουν πιθανότητα γύρω στο 50 το εκάστοτε λογισμικό αδυνατεί να αποφασίσει αν αυτά είναι επιθυμητά η ανεπιθύμητα. Στα διαγράμματα οι διάφορες πιθανότητες 0-100 εμφανίζονται στον άξονα χ ενώ το ποσοστό μηνυμάτων που αντιστοιχεί στην κάθε πιθανότητα στον άξονα y.

Για τον υπολογισμό των σφαλμάτων θα υποθέσουμε ότι τα μηνύματα με πιθανότητες 0-55% θεωρούνται επιθυμητά ενώ τα μηνύματα με πιθανότητες 55-100% θεωρούνται ανεπιθύμητα.

4.1.2.1 Spamassassin





Στην δεύτερη μέτρηση το Spamassassin δεν αποδίδει καλά στον διαχωρισμό των επιθυμητών μηνυμάτων. Το πρόγραμμα αδυνατεί να κατατάξει τα περισσότερα από αυτά σε κάποια κατηγορία με αποτέλεσμα να παρουσιάζεται μεγάλη συγκέντρωση γύρω από την πιθανότητα 50%. Τα υπόλοιπα μυνήματα εμφανίζονται διασκορπισμένα σε όλες σχεδόν τις πιθανότητες. Η πιθανότητα σφάλματος False Positive είναι P_{FP}=0.132

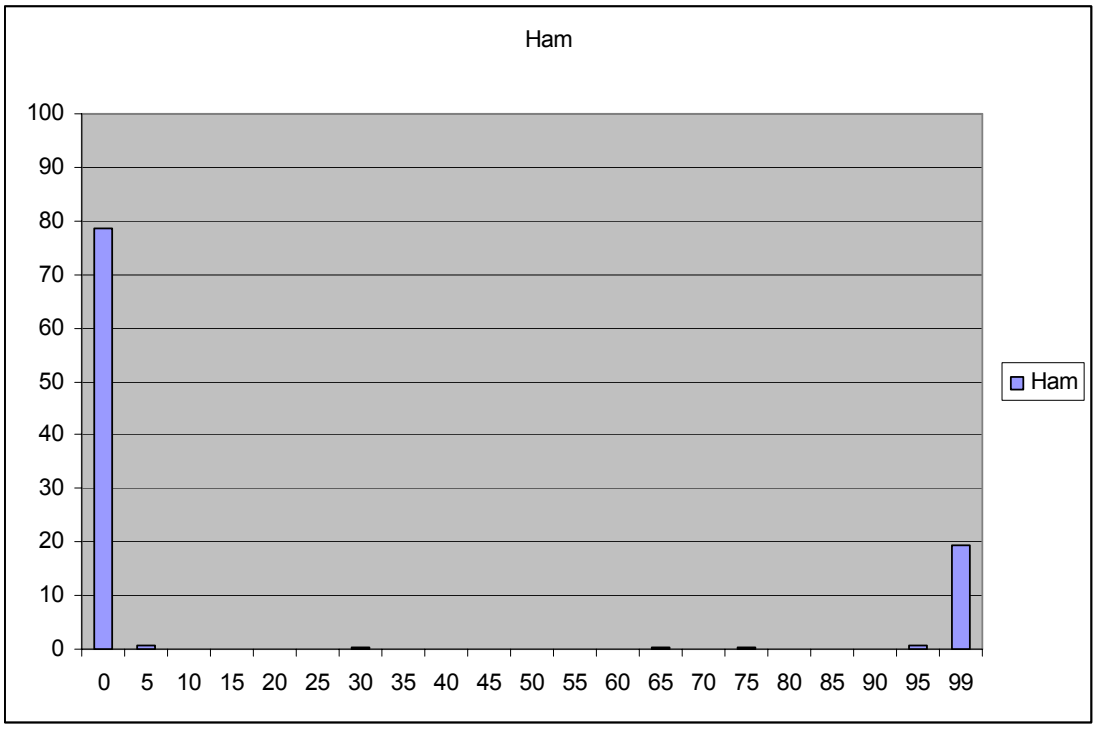
Στην περίπτωση διαχωρισμού των ανεπιθύμητων μηνυμάτων το spamassassin αποδίδει αρκετά καλά. Σε σχέση με την μέτρηση 1 βλέπουμε μια αύξηση του ποσοστού των μηνυμάτων που το Spamassassin αδυνατεί ουσιαστικά να διαχωρίσει (αυτά που εμφανίζουν πιθανότητα γύρω στο 50%). Το σφάλμα False Negative είναι P_{FN}= 0.112.

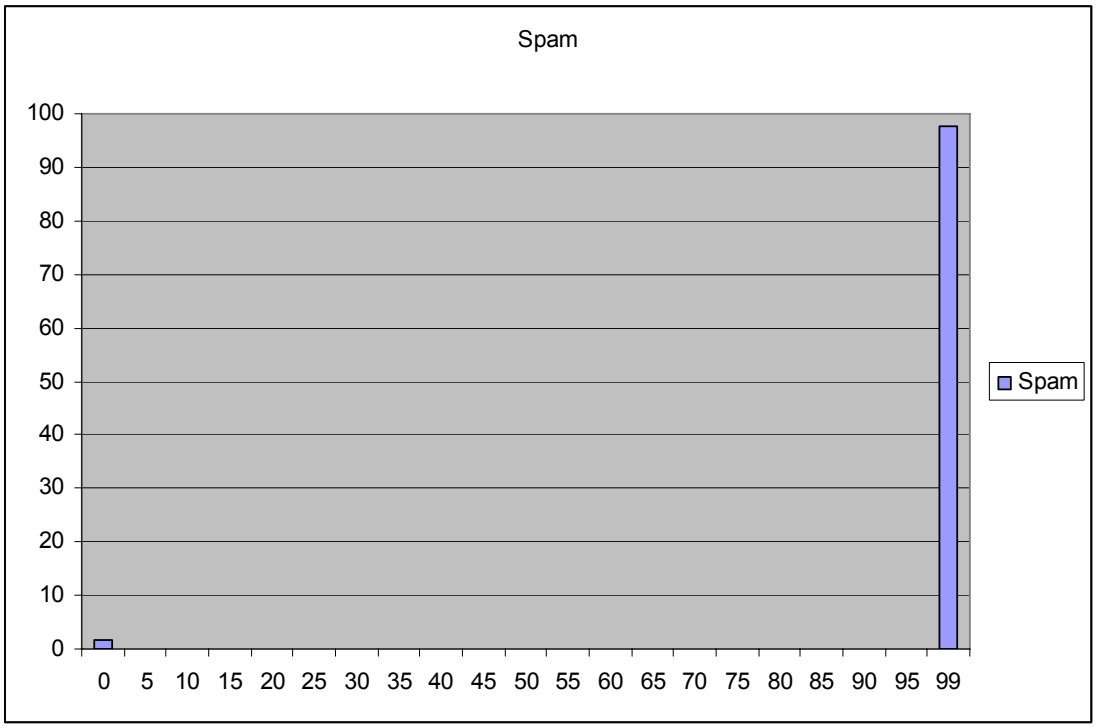
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.12$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.13$

Παρατηρούμε ότι τα σφάλματα έχουν αυξηθεί πάρα πολύ, περίπου 1000% για την περίπτωση του λ=9. Πράγματι στην μέτρηση 2 η απόδοση του spamassassin στον διαχωρισμό των επιθυμητών μηνυμάτων έχει μειωθεί αισθητά. Ας προχωρήσουμε όμως και στα υπόλοιπα προγράμματα.

4.1.2.2 Dspam


Ham
100
90
80
70
60
50
40
30
20
10
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 99
Ham


Spam
100
90
80
70
60
50
40
30
20
10
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 99
Spam

Από τα διαγράμματα του Dspam παρατηρούμε ότι όπως και στην μέτρηση 1 υπάρχει μεγάλη πόλωση των αποτελεσμάτων. Πράγματι κατατάσσει την πλειοψηφία των μηνυμάτων είτε ως σίγουρα spam (πιθανότητα 100%) είτε ως σίγουρα ham (πιθανότητα 0%)

Από το διάγραμμα διαχωρισμού των επιθυμητών μηνυμάτων βλέπουμε ότι το dspam κατατάσσει σωστά ένα αρκετά μεγάλο ποσοστό των μηνυμάτων ως επιθυμητά. Το ποσοστό αυτό είναι περίπου 80%. Ωστόσο τα επιθυμητά μηνύματα που αποτυγχάνει να διαχωρίσει σωστά κατατάσσονται ως σίγουρα spam. Ο αριθμός αυτών των μηνυμάτων είναι αυξημένος σε σχέση με την μέτρηση 1 και πλησιάζει το 20%. Το σφάλμα False Positives είναι P_{FP}=0.21

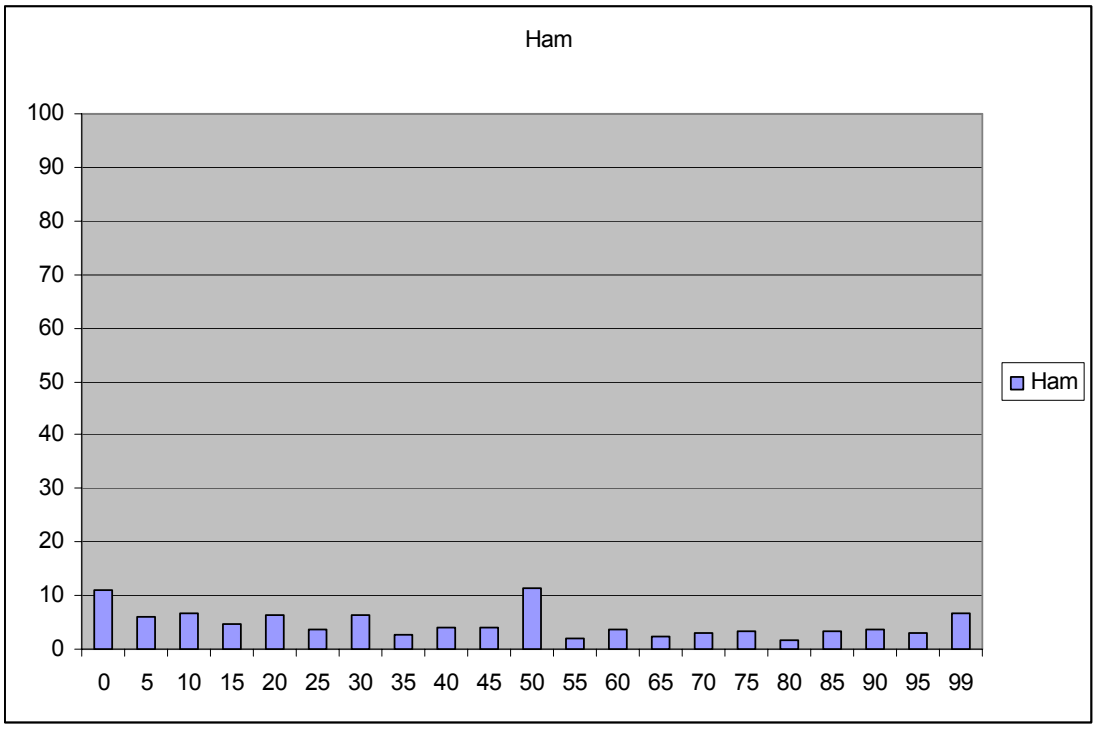
Αντίθετα στο διάγραμμα των ανεπιθύμητων μηνυμάτων παρατηρούμε μια μικρή βελτίωση της απόδοσης σε σχέση με την μέτρηση 1. Το dspam μπορεί και κατατάσσει σωστά ένα μεγάλο ποσοστό των ανεπιθύμητων μηνυμάτων. Το σφάλμα False Negatives είναι για την περίπτωση του dspam P_{FN}=0.02

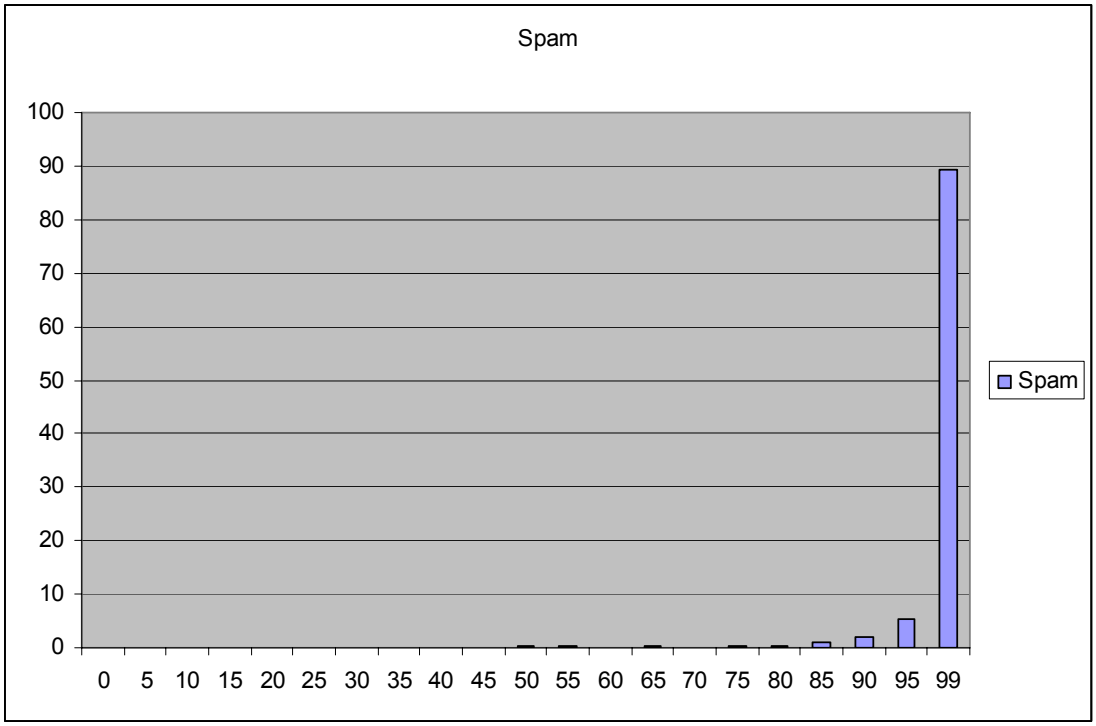
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.11$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.19$

Παρατηρούμε μια σημαντική αύξηση του σφάλματος σε σχέση με την μέτρηση 1, ειδικά στην περίπτωση όπου λ=9. Αυτό οφείλεται κυρίως στην μείωση της ικανότητας διαχωρισμού των επιθυμητών μηνυμάτων από το πρόγραμμα αφού για τα ανεπιθύμητα η απόδοση έχει μείνει σταθερή.

4.1.2.3 Spambayes





Από το γράφημα της πρώτης μέτρησης μπορούμε να συμπεράνουμε ότι το Spambayes αδυνατεί σε μεγάλο βαθμό να πετύχει σωστό διαχωρισμό των επιθυμητών μηνυμάτων αφού υπάρχει σχεδόν ομοιόμορφη κατανομή των μηνυμάτων σε όλες τις πιθανότητες από 0% εώς 100%. Παρατηρούμε επίσης μια μικρή συγκέντρωση στις συχνότητες 0% και 50%. Το σφάλμα False Positive είναι P_{FP}=0.33 που σημαίνει ότι ένα στα τρία επιθυμητά μηνύματα κατατάσσεται ως ανεπιθύμητο από το Spambayes.

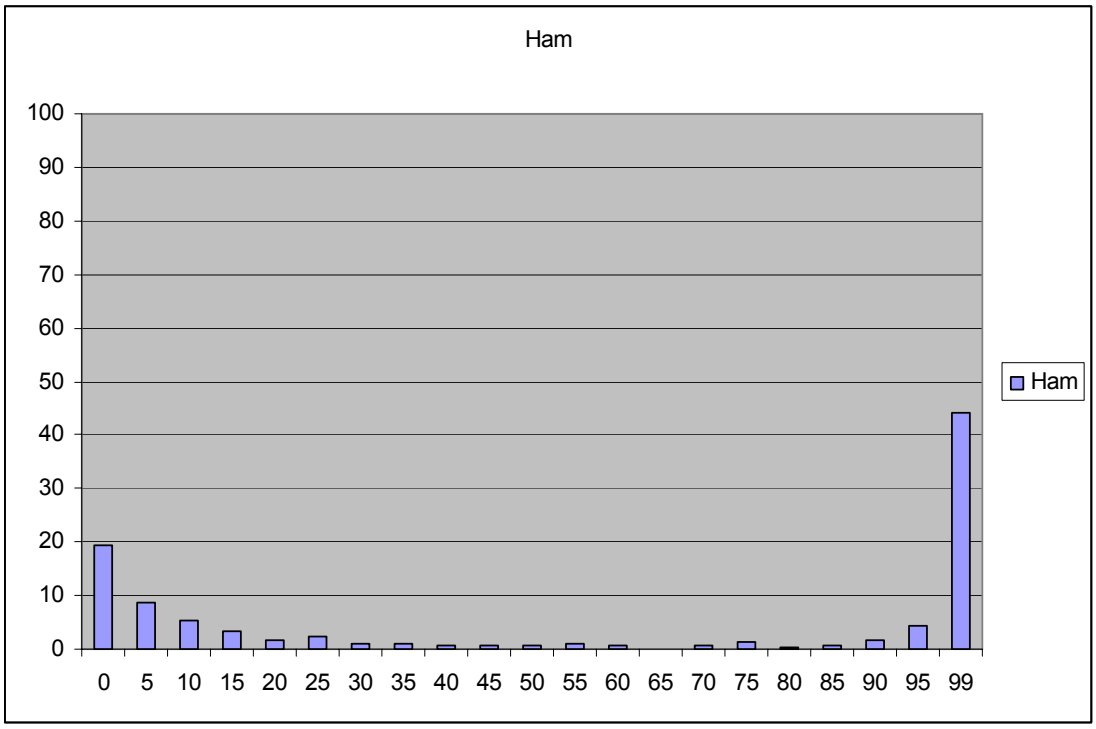
Στην περίπτωση διαχωρισμού των ανεπιθύμητων μηνυμάτων η εικόνα είναι τελείως διαφορετική από αυτή των επιθυμητών μηνυμάτων. Τα περισσότερα αναγνωρίζονται με πλήρη εμπιστοσύνη ως spam από το πρόγραμμα. Τα υπόλοιπα μηνύματα που το spambayes είτε δυσκολεύται να κατατάξει είτε τα κατατάσσει λάθος μοιράζονται σε διάφορες πιθανότητες χωρίς να παρουσιάζεται ιδιαίτερη συγκέντρωση ούτε στην περιοχή των μηνυμάτων που αδυνατεί να διαχωρίσει (κοντά στο 50%) ούτε στην περιοχή των μηνυμάτων που κατατάσσονται εντελώς εσφαλμένα ως επιθυμητά (κοντά στο 0%). Το σφάλμα False Negative είναι P_{FN}=0.01 το ίδιο με αυτό της μέτρησης 1.

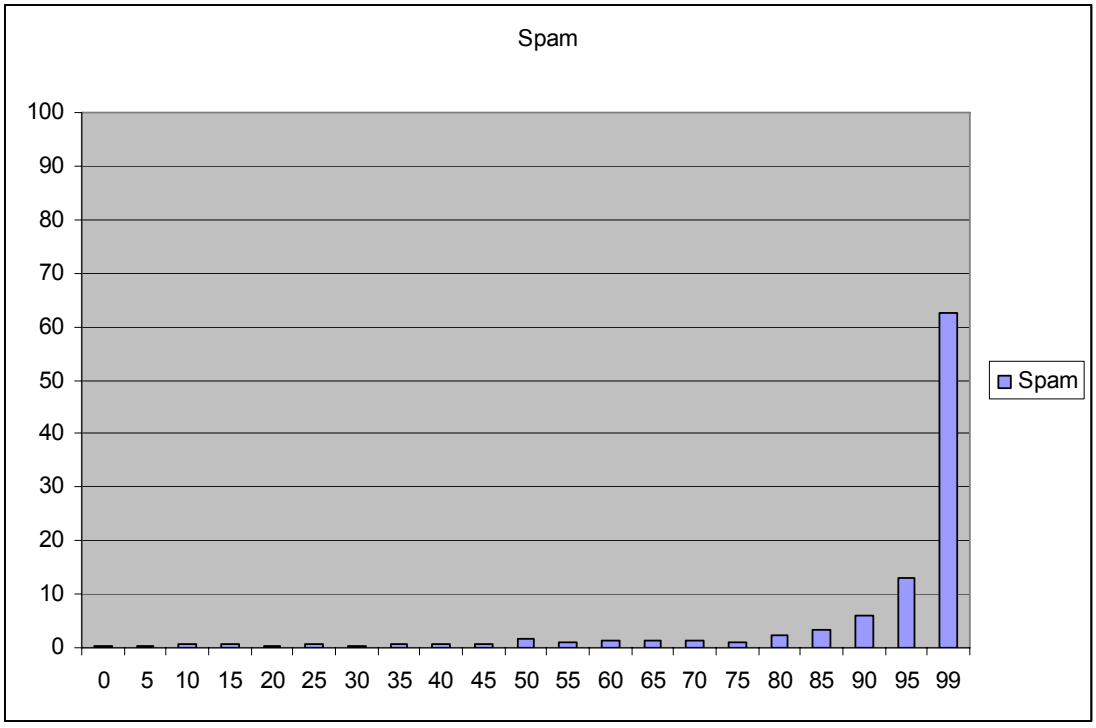
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.17$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.30$

Παρατηρούμε ότι το σφάλμα και για τις δυο περιπτώσεις του λ είναι σημαντικά αυξημένο σε σχέση με την μέτρηση 1. Η πολύ καλή απόδοση στην ανίχνευση των ανεπιθύμητων μηνυμάτων επισκιάζεται από το μεγάλο σφάλμα που εμφανίζεται στην ανίχνευση των επιθυμητών μηνυμάτων για το spambayes.

4.1.2.4 Spamprobe





Από το πρώτο διάγραμμα παρατηρούμε την πολύ άσχημη απόδοση του spamprobe στην κατάταξη των επιθυμητών μηνυμάτων. Πράγματι η πλειοψηφία κατατάσσεται λάθος ως ανεπιθύμητα μηνύματα με αρκετά μεγάλη συγκέντρωση στην περιοχή 100%. Για τα μηνύματα αυτά δηλαδή το spamprobe, σε αντίθεση με την πραγματικότητα, είναι βέβαιο ότι είναι ανεπιθύμητα. Δεν παρατηρούμε συγκέντρωση μηνυμάτων στην περιοχή 50% που αφορά μηνύματα που το πρόγραμμα δυσκολεύεται να κατατάξει. Έτσι το σφάλμα False Positive για την περίπτωση του Spamprobe παίρνει την πολύ μεγάλη τιμή P_{FP}=0.55 , δηλαδή περισσότερα από τα μισά μηνύματα κατατάσσονται λάθος.

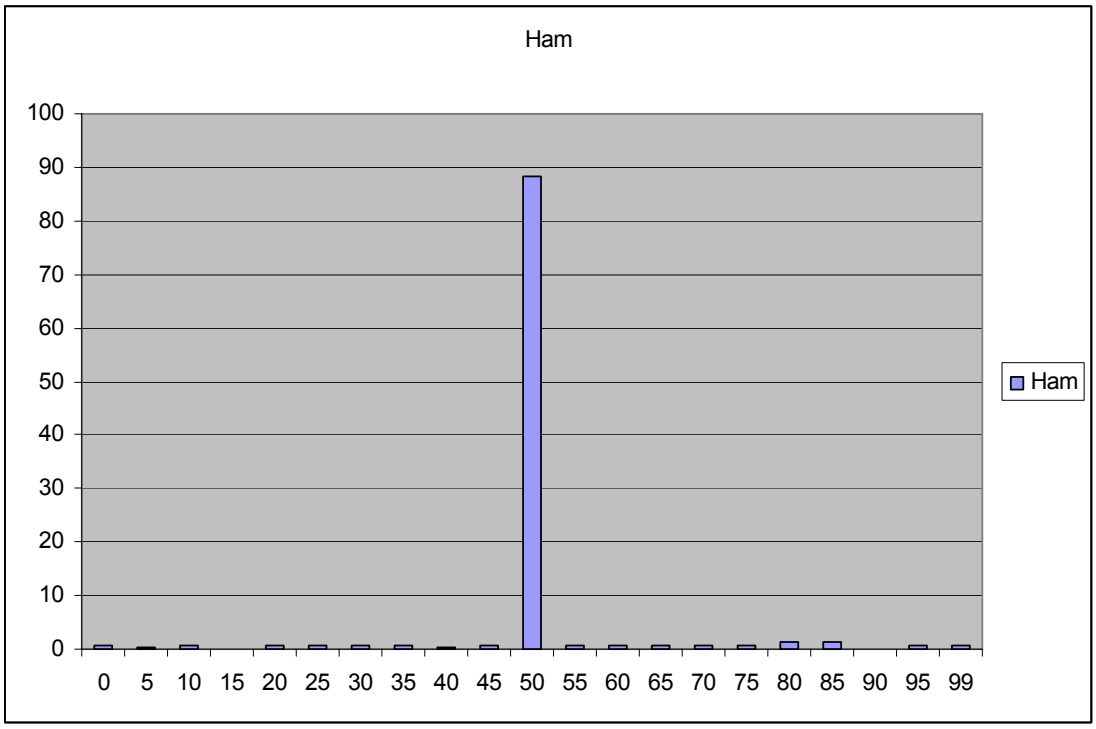
Στην περίπτωση κατάταξης των ανεπιθύμητων μηνυμάτων παρατηρούμε μια διαφορετική εικόνα από αυτή των επιθυμητών. Το πρόγραμμα κατατάσσει με πλήρη εμπιστοσύνη ως ανεπιθύμητα ένα ικανό ποσοστό (περίπου 65%) των μηνυμάτων. Πολλά από τα υπόλοιπα μηνύματα βρίσκονται διασκορπισμένα σε όλο το φάσμα των πιθανοτήτων είτε με σωστή είτε με λάθος κατάταξη. Το σφάλμα False Negative είναι P_{FN}=0.07, πολύ μικρότερο σε σχέση με το σφάλμα False Positive.

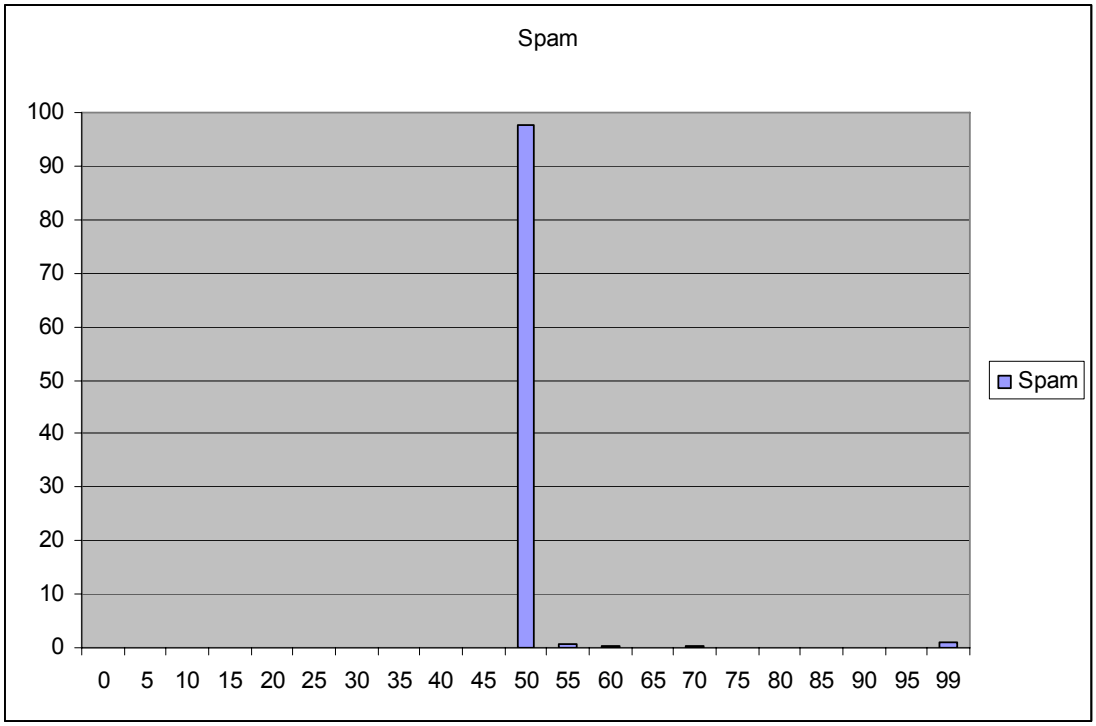
Το συνολικό σφάλμα είναι:

- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.31$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.50$

Και στις δυο περιπτώσεις του λ το σφάλμα είναι εξαιρετικά μεγάλο.

4.1.2.5 Bogofilter





Από τα διαγράμματα του bogofilter παρατηρούμε μια διαφορετική εικόνα από αυτήν που παρουσίασαν τα υπόλοιπα προγράμματα στην μέτρηση 2. Πράγματι το bogofilter φαίνεται να αδυνατεί να πετύχει κάποιον διαχωρισμό το μυνημάτων είτε στην περίπτωση που πρόκειται για επιθυμητά είτε για ανεπιθύμητα μηνύματα. Και από τα δυο διαγράμματα παρατηρούμε μεγάλη συγκέντρωση των μηνυμάτων στην περιοχή της πιθανότητας 50%. Ιδιαίτερα στην περίπτωση των ανεπιθύμητων μηνυμάτων η συγκέντρωση φτάνει περίπου το 98% των μηνυμάτων. Τα υπόλοιπα μηνύματα είναι διασκορπισμένα σε διάφορες πιθανότητες. Και τα υπόλοιπα προγράμματα εμφάνισαν δυσκολίες στην μέτρηση αυτή, το bogofilter όμως αδυνατεί τελείως να πετύχει κάποιο διαχωρισμό στα μηνύματα είτε πρόκειται για επιθυμητά είτε για ανεπιθύμητα. Ως προς την αιτία εμφάνισης αυτής της απόδοσης από το bogofilter ο γράφων δεν μπορεί να εκφέρει κάποια άποψη, δεδομένου ότι διαφέρει αρκετά από την εικόνα που παρουσιάζουν τα υπόλοιπα προγράμματα της δοκιμής.

Το σφάλμα False Positive είναι P_{FP}=0.07 ενώ το σφάλμα False Negative είναι P_{FN}=0.98

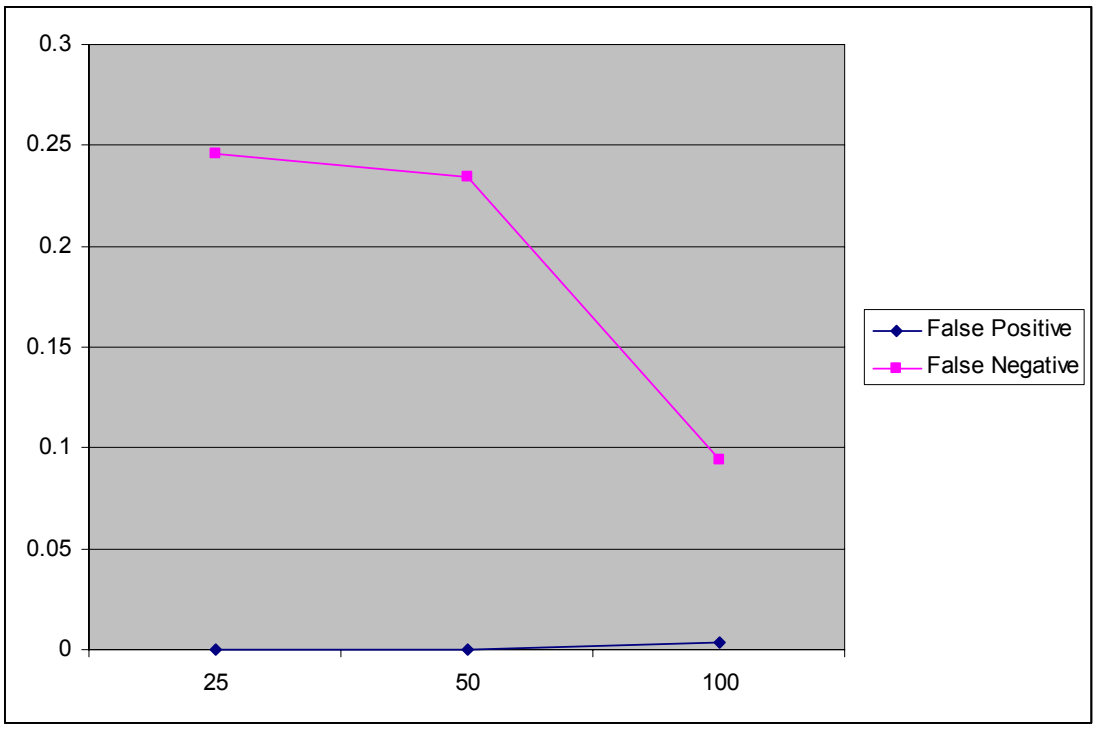
Το συνολικό σφάλμα είναι:

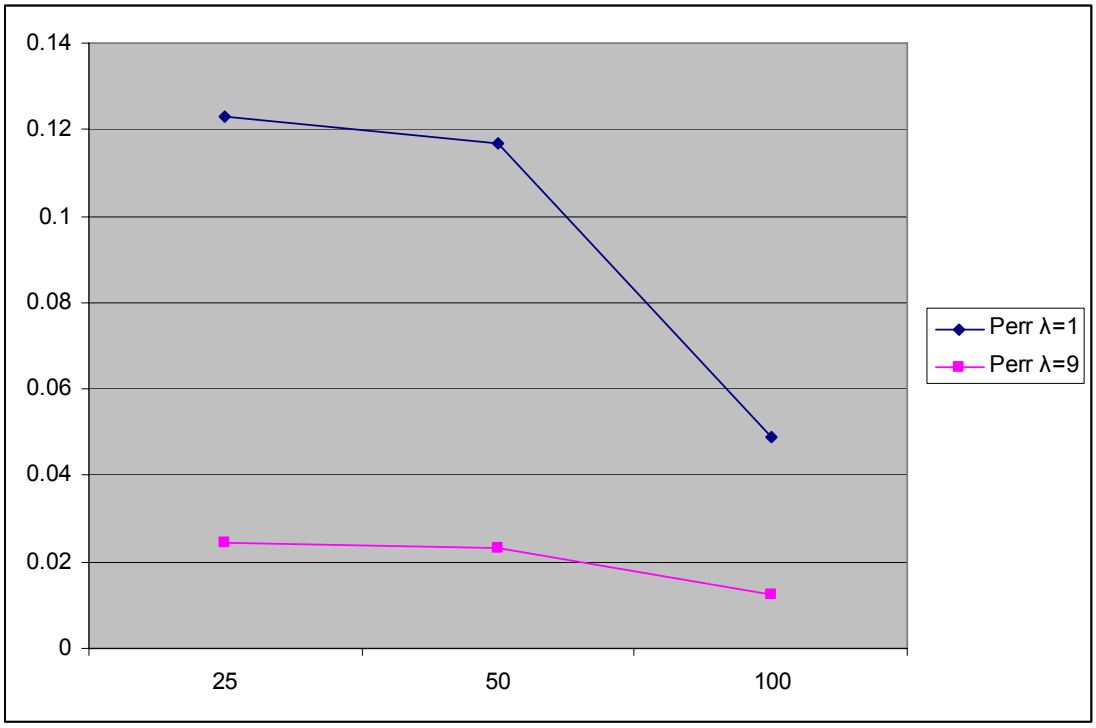
- Για λ=1: $P_{err} = \frac{P_{FP} + P_{FN}}{2} = 0.52$
- Για λ=9: $P_{err} = \frac{\lambda \cdot P_{FP} + P_{FN}}{\lambda + 1} = 0.16$

Το σφάλμα False Positive και το συνολικό για λ=9 εμφανίζονται αρκετά μικρά, αφού έχουμε επιλέξει ότι τα μηνύματα με πιθανότητα 50% κατατάσσονται ως επιθυμητά για τις μετρησεις που έγιναν, ωστώσω στην πραγματικότητα για την μέτρηση 2 η απόδοση του bogofilter είναι η χειρότερη από τα προγράμματα που εξετάστηκαν.

4.1.3 Μέτρηση 3

4.1.3.1 Spamassassin





Στο πρώτο διάγραμμα για το spamassassin βλέπουμε μια αναντιστοιχία στην απόδοση ανάμεσα στην ανίχνευση των επιθυμητών και των ανεπιθύμητων μηνυμάτων. Πράγματι το πρόγραμμα αποδίδει πολύ καλύτερα στην ανίχνευση των επιθυμητών ακόμα και όταν χρησιμοποιήθηκε μόλις το 25% των μηνυμάτων για εκπαίδευση και το σφάλμα False Positive βρίσκεται συνεχώς κοντά στο 0 και για τα τρία μεγέθη του dataset εκπαίδευσης.

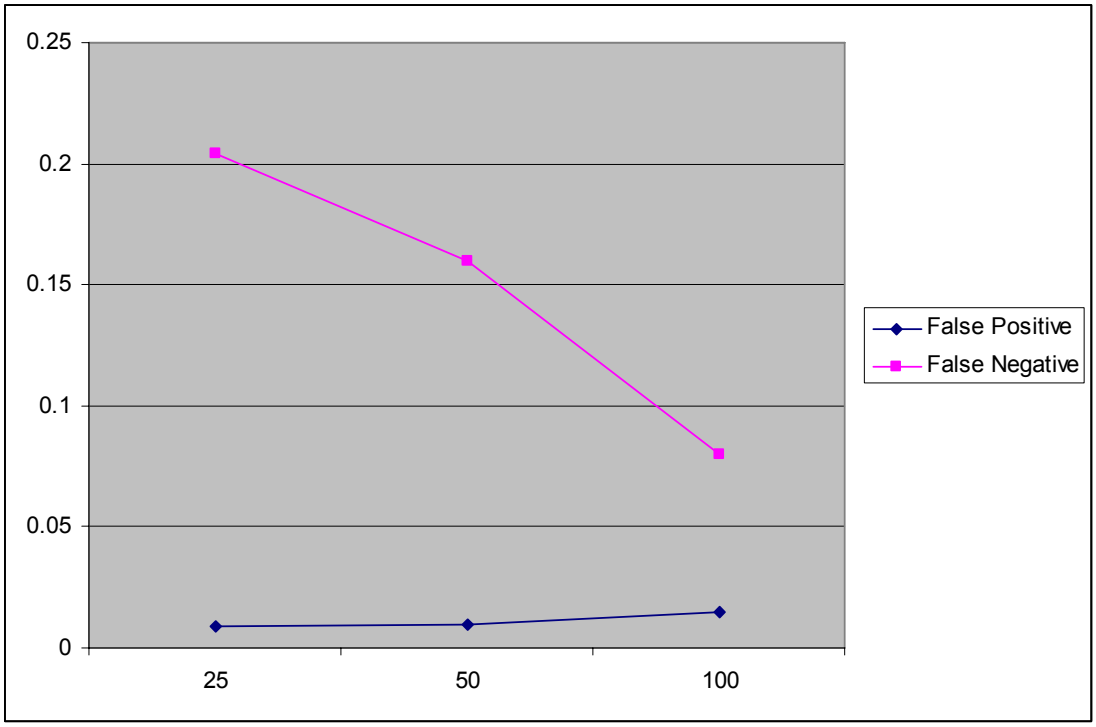
Αντίθετα στην περίπτωση ανίχνευσης των ανεπιθύμητων μηνυμάτων το spamassassin ξεκινά με αρκετά άσχημη απόδοση για το μικρό dataset, P_{FN}=0,246, και βελτιώνεται μέχρι το P_{FN}=0,094 για το πλήρες dataset.

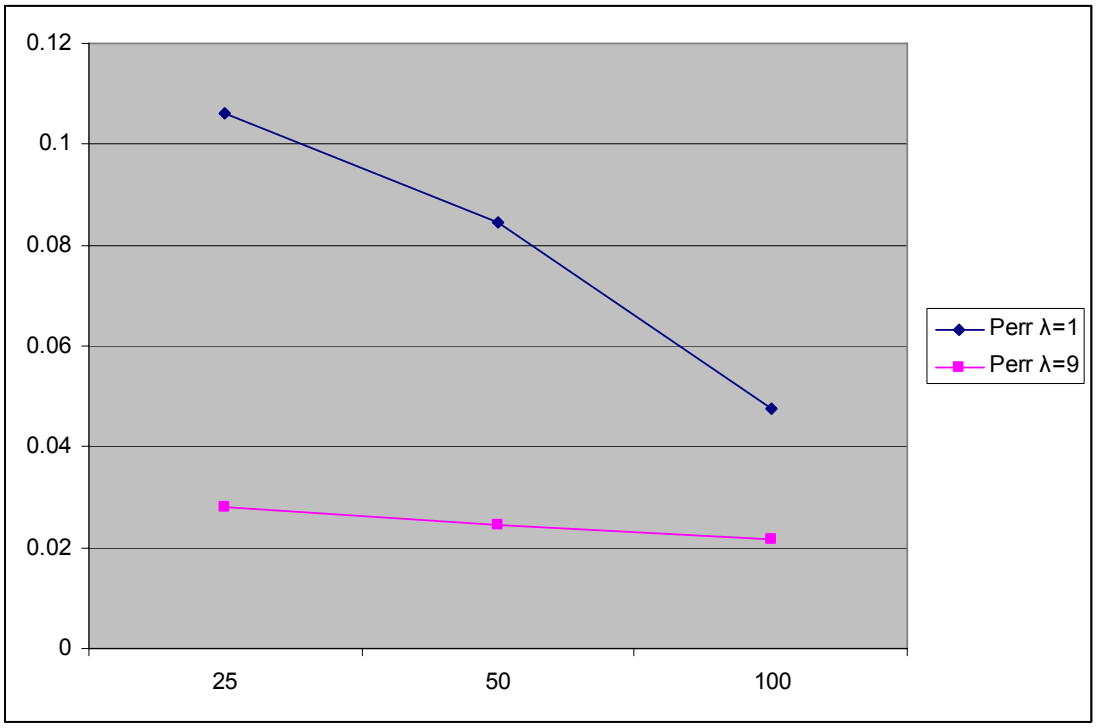
Αντοίστοιχη εικόνα έχουμε και για τα δυο συνολικά σφάλματα. Πράγματι για λ=1 το σφάλμα P_{err} παίρνει αρκετά μεγάλες τιμές για μικρό dataset εκμάθησης και μειώνεται στην συνέχεια, όσο αυξάνεται το πλήθος των μηνυμάτων του dataset εκμάθησης. Για λ=9 το συνολικό σφάλμα P_{err} ακολουθεί και αυτό φθίνουσα πορεία όσο αυξάνεται το πλήθος των μηνυμάτων εκμάθησης. Ωστόσο οι τιμές του γενικά είναι αρκετά μικρότερες από το σφάλμα για λ=1 κάτι που οφείλεται κυρίως στην μεγάλη διαφορά μεταξύ των σφαλμάτων False Positive και False Negative. Το συνολικό σφάλμα για λ=9 βασίζεται κυρίως στο σφάλμα False Positive.

Συμπερασματικά μπορούμε να πούμε ότι το spamassassin όσων αφορά τα επιθυμητά μηνύματα τα διαχωρίζει σωστά ακόμα και για μικρά dataset εκμάθησης, ενώ για τα ανεπιθύμητα μηνύματα χρειάζεται όσο το δυνατόν μεγαλύτερο dataset εκμάθησης για να αποδώσει σωστά.

Το θετικό αυτής της συμπεριφοράς είναι ότι ακόμη κι αν δεν υπάρχουν πολλά δεδομένα για εκμάθηση μπορεί να αρχίσει κάποιος να χρησιμοποιεί το πρόγραμμα χωρίς τον φόβο ότι θα χάσει κάποια επιθυμητά μηνύματα. Βέβαια θα κατατάσσονται λάθος αρκετά ανεπιθύμητα μηνύματα.

4.1.3.2 Dspam


0.25
0.2
0.15
0.1
0.05
0
25
50
100
False Positive
False Negative


0.12
0.1
0.08
0.06
0.04
0.02
0
25
50
100
Perr λ=1
Perr λ=9

Στο πρώτο διάγραμμα για το dspam παρατηρούμε απόδοση αντίστοιχη με το spamassassin. Πράγματι υπάρχει μια αναντιστοιχία στην απόδοση ανάμεσα στην ανίχνευση των επιθυμητών και των ανεπιθύμητων μηνυμάτων. Το πρόγραμμα αποδίδει καλύτερα στην ανίχνευση των επιθυμητών ακόμα και όταν χρησιμοποιήθηκε μόλις το 25% των μηνυμάτων για εκπαίδευση. Επίσης ενδιαφέρον δημιουργεί το γεγονός ότι στην περίπτωση αυτή η απόδοση είναι καλύτερη για το dataset 25% σε σχέση με το dataset 100%, χωρίς βέβαια η διαφορά να είναι μεγάλη.

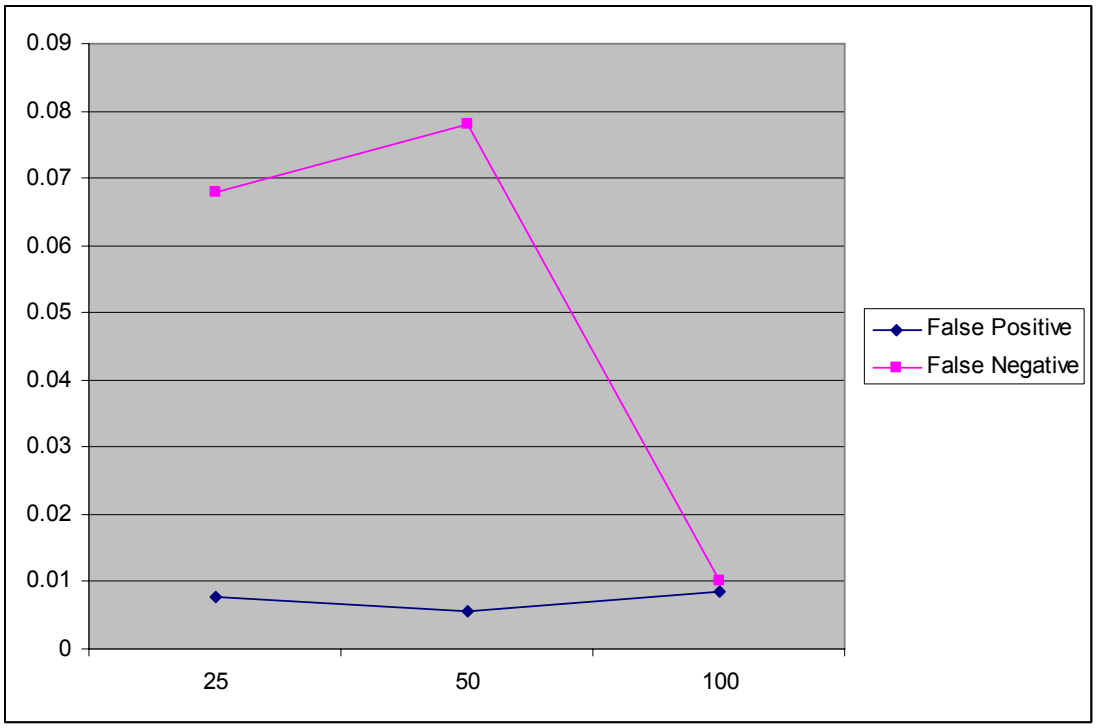
Στην περίπτωση ανίχνευσης των ανεπιθύμητων μηνυμάτων το dspam ξεκινά με άσχημη απόδοση για το μικρό dataset, P_{FN}=0,20, και βελτιώνεται αισθητά στην συνέχεια μέχρι το P_{FN}=0,08 για το πλήρες dataset.

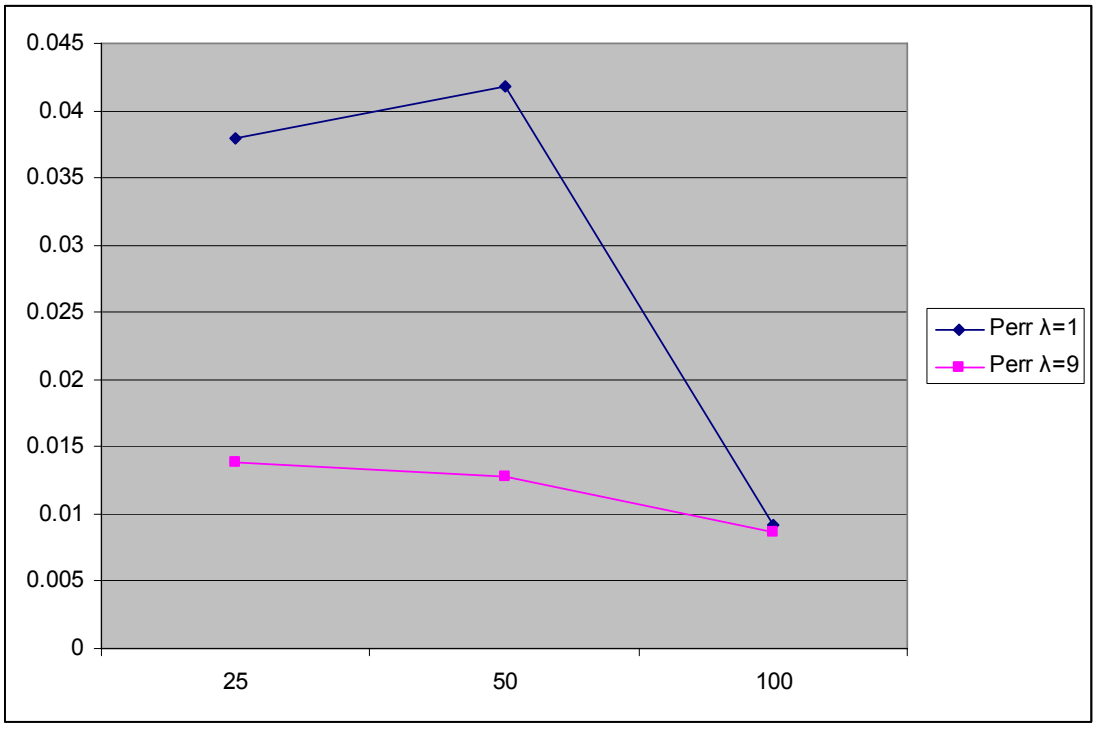
Όσον αφορά τα δυο συνολικά σφάλματα για λ=1 το σφάλμα P_{err} παίρνει αρκετά μεγάλες τιμές για μικρό dataset εκμάθησης και μειώνεται στην συνέχεια, όσο αυξάνεται το πλήθος των μηνυμάτων του dataset εκμάθησης. Για λ=9 το συνολικό σφάλμα P_{err} ακολουθεί και αυτό φθίνουσα πορεία όσο αυξάνεται το πλήθος των μηνυμάτων εκμάθησης και είναι γενικά μικρότερο από το σφάλμα για λ=1.

Το dspam όπως και το spamassassin διαχωρίζει σωστά τα επιθυμητά μηνύματα ακόμα και για μικρά dataset εκμάθησης, ενώ για τα ανεπιθύμητα μηνύματα χρειάζεται όσο το δυνατόν μεγαλύτερο dataset εκμάθησης για να αποδώσει σωστά.

Μάλιστα σε σύγκριση με το spamassassin το dspam φαίνεται να είναι πιο ευαίσθητο στο μέγεθος του dataset εκμάθησης αφού η απόδοσή του στην ανίχνευση των ανεπιθύμητων μηνυμάτων αυξάνεται σημαντικά όσο μεγαλύτερο είναι το dataset.

4.1.3.3 Spambayes





Στο πρώτο διάγραμμα του spambayes υπάρχει, όπως και στα προηγούμενα προγράμματα, διαφορά στην απόδοση ανάμεσα στην ανίχνευση των επιθυμητών και των ανεπιθύμητων μηνυμάτων. Η ανίχνευση των επιθυμητών γίνεται σωστά ακόμα και όταν χρησιμοποιήθηκε μόλις το 25% των μηνυμάτων για εκπαίδευση. Στην συνέχεια η απόδοση βελτιώνεται λίγο για το dataset 50% και ελαττώνεται πάλι στο dataset 100%, χωρίς βέβαια η διαφορά να είναι μεγάλη.

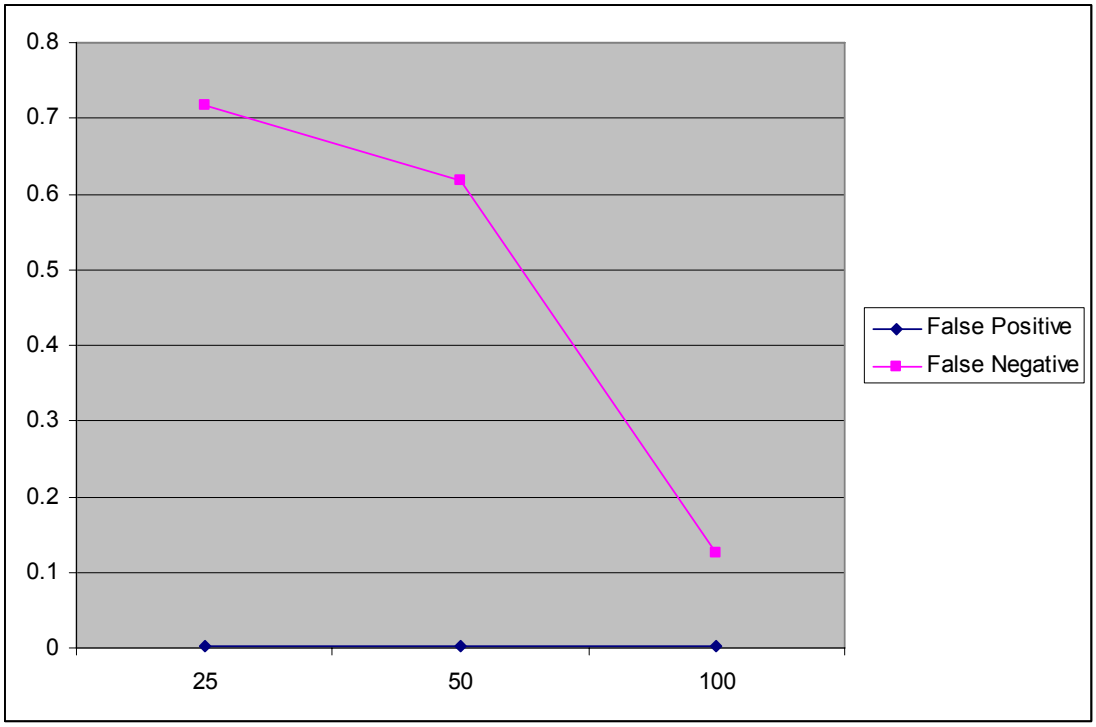
Στην περίπτωση ανίχνευσης των ανεπιθύμητων μηνυμάτων παρατηρούμε μια εικόνα διαφορετική από αυτή των προηγούμενων προγραμμάτων που εξετάστηκαν. Το spambayes ξεκινά με σφάλμα P_{FN}=0,068 για το μικρό dataset. Στην συνέχεια το σφάλμα γίνεται αισθητά μεγαλύτερο για το dataset 50% P_{FN}=0,078. Τέλος παίρνει την ελάχιστη τιμή του για το πλήρες dataset P_{FN}=0.01, που είναι και η καλύτερη απόδοση μεταξύ των προγραμμάτων που εξετάσθηκαν.

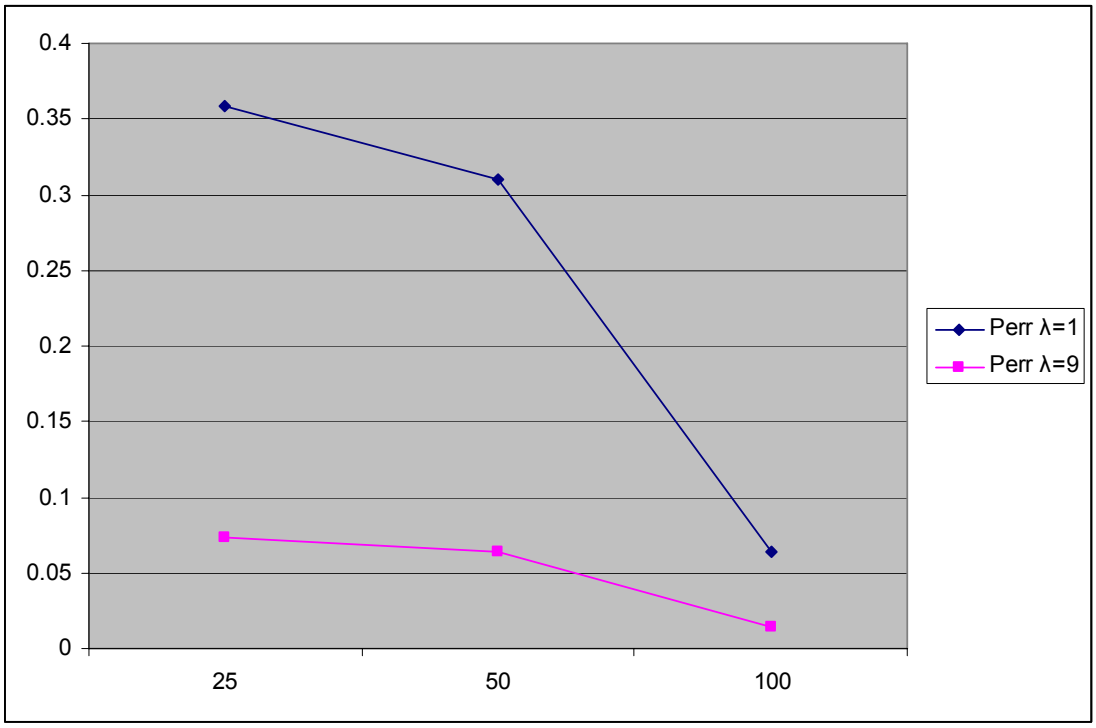
Όσον αφορά τα δυο συνολικά σφάλματα για λ=1 το σφάλμα P_{err} παίρνει αρκετά μεγάλες τιμές για το μικρό, 25%, και το μεσαίο, 50%, dataset εκμάθησης και μειώνεται σε μεγάλο βαθμό για το πλήρες dataset. Για λ=9 το συνολικό σφάλμα P_{err} ακολουθεί φθίνουσα πορεία όσο αυξάνεται το πλήθος των μηνυμάτων εκμάθησης και είναι γενικά μικρότερο από το σφάλμα για λ=1, εκτός από την μέτρηση για το πλήρες dataset όπου τα δυο σφάλματα παίρνουν την ίδια περίπου τιμή.

Το spambayes όπως και τα προηγούμενα προγράμματα διαχωρίζει σωστά τα επιθυμητά μηνύματα ακόμα και για μικρά dataset εκμάθησης, ενώ για τα ανεπιθύμητα μηνύματα χρειάζεται όσο το δυνατόν μεγαλύτερο dataset εκμάθησης για να αποδώσει σωστά.

Θα πρέπει να προσθέσουμε εδώ ότι η απόδοση του spambayes στην μέτρηση αυτή ήταν η καλύτερη από τα προγράμματα που εξετάσαμε. Ειδικά στο πλήρες dataset όλα τα σφάλματα παίρνουν πολύ μικρή τιμή.

4.1.3.4 Spamprobe





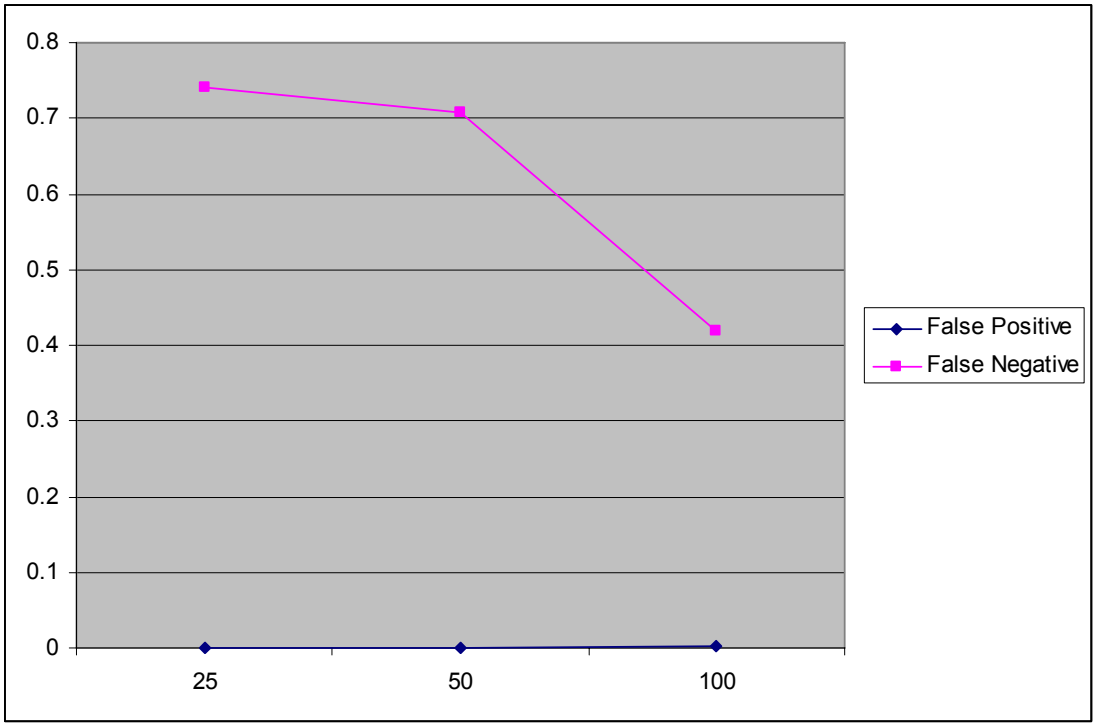
Στο πρώτο διάγραμμα του spamprobe εμφανίζεται η τυπική, όπως και στα υπόλοιπα προγράμματα, διαφορά μεταξύ της απόδοσης των επιθυμητών και ανεπιθύμητων μηνυμάτων. Η ανίχνευση των επιθυμητών στην περίπτωση του spamprobe είναι άριστη αφού ακόμη και για το μικρότερο dataset 25% το σφάλμα False Positive είναι σχεδόν μηδενικό P_{FP}=0,001. Η άριστη αυτή απόδοση διατηρείται και στα υπόλοιπα dataset.

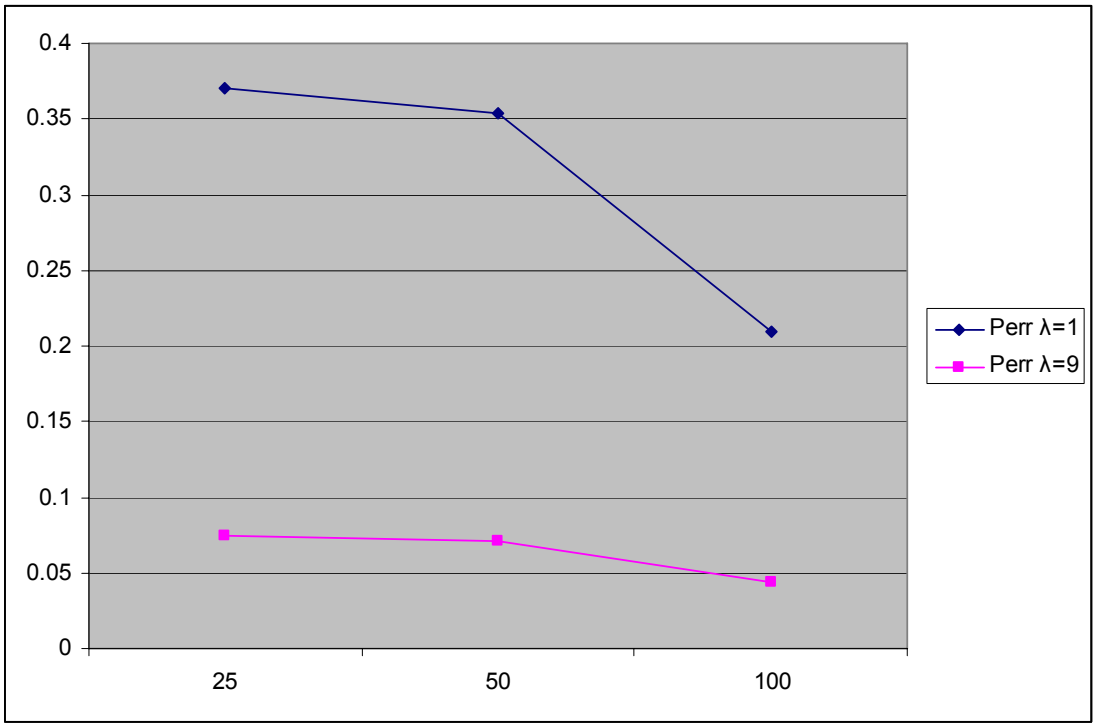
Στην περίπτωση ανίχνευσης των ανεπιθύμητων μηνυμάτων η τα σφάλματα είναι πολύ μεγάλα ειδικά για την περίπτωση του μικρού και του μεσαίου dataset όπου παίρνουν τις τιμες P_{FN}=0,72 και P_{FN}=0,62 αντίστοιχα. Μια τέτοια απόδοση δεν μπορεί να είναι αποδεκτή σε πραγματική λειτουργία του προγράμματος. Στο μεγαλύτερο dataset το σφάλμα παίρνει την ελάχιστη τιμή του P_{FN}=0.12, η οποία είναι μέτρια αλλά σίγουρα πολύ καλύτερη από αυτή των μικρότερων dataset.

Όσον αφορά τα δυο συνολικά σφάλματα για λ=1 το σφάλμα P_{err} παίρνει μεγάλες τιμές για το μικρό, 25%, και το μεσαίο, 50%, dataset εκμάθησης και μειώνεται αρκετά στο πλήρες dataset. Για λ=9 η απόδοση είναι καλύτερη σε σχέση με το λ=1.

Η απόδοση του spamprobe στην μέτρηση αυτή ήταν πολύ άσχημη, ειδικά για τα μικρά μεγέθη των dataset, αφού περισσότερα από τα μισά ανεπιθύμητα μηνύματα κατατάσσονταν λάθος. Στο πλήρες dataset η απόδοση του προγράμματος ήταν αρκετά βελτιωμένη σε σχέση με τα μικρότερα dataset χωρίς όμως να ξεπερνά το μέτριο.

4.1.3.5 Bogofilter





Από το διάγραμμα του bogofilter, του τελευταίου προγράμματος που εξετάζουμε, παρατηρούμε ότι για την περίπτωση διαχωρισμού των επιθυμητών μηνυμάτων η απόδοση του είναι πολύ καλή αφού και για τα τρία μεγέθη dataset το σφάλμα False Positive βρίσκεται κοντά στο μηδέν.

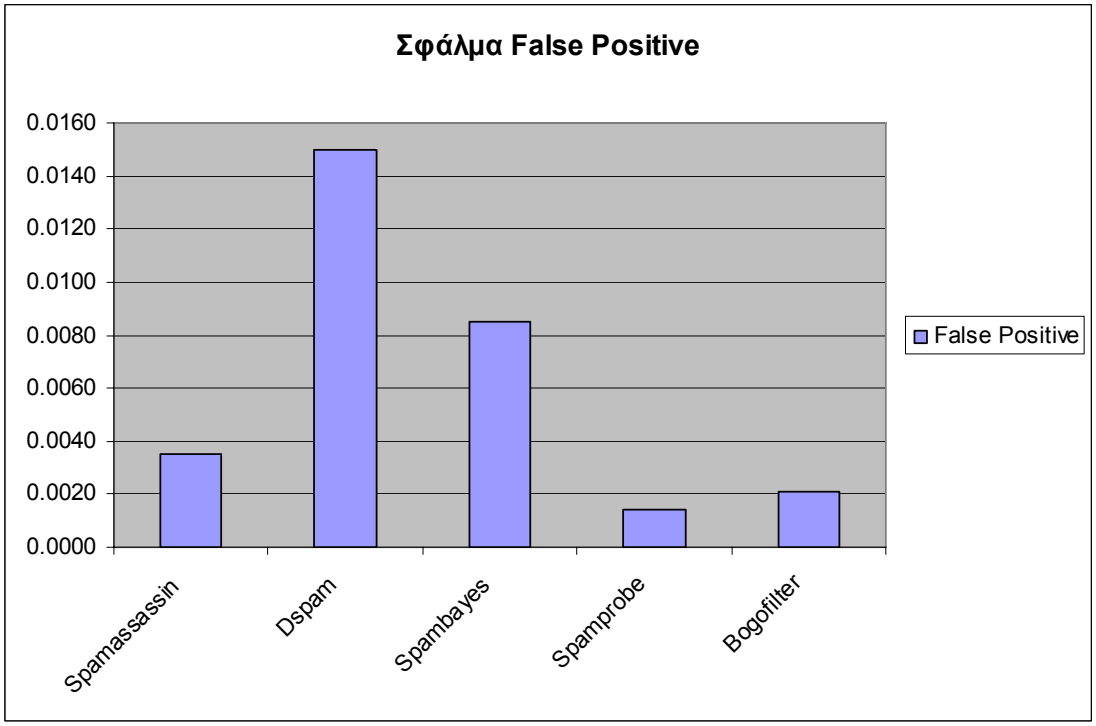
Στην περίπτωση όμως διαχωρισμού των ανεπιθύμητων μηνυμάτων η απόδοση του προγράμματος είναι πολύ άσχημη, είναι μάλιστα η χειρότερη από τα προγράμματα που εξετάσθηκαν. Στα δυο μικρότερα dataset το σφάλμα False Negative παίρνει τις τιμές P_{FN}=0,74 και P_{FN}=0,70, δηλαδή σχεδόν 3 στα 4 ανεπιθύμητα μηνύματα κατατάσσονται λάθος ως επιθυμητά. Στο πλήρες dataset η απόδοση του προγράμματος βελτιώνεται αλλά παραμένει μη αποδεκτή και το σφάλμα είναι P_{FN}=0,41.

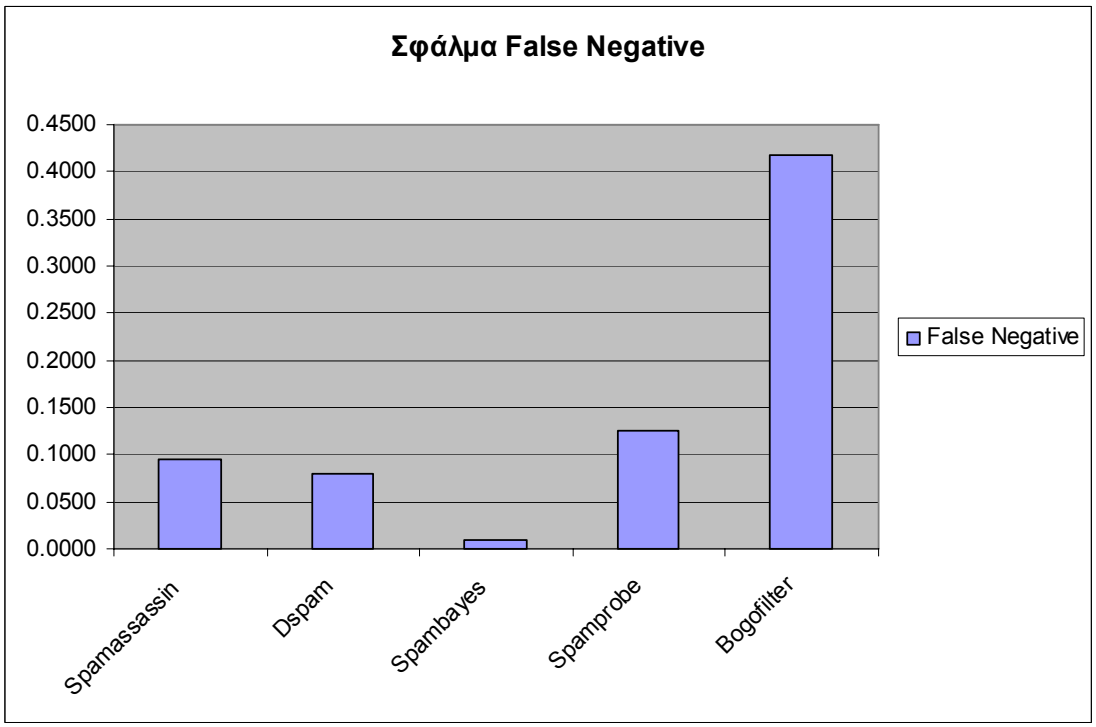
Στην περίπτωση του συνολικού σφάλματος για λ=1 και για λ=9 η εικόνα παραμένει άσχημη και το bogofilter είναι με διαφορά το χειρότερο από τα προγράμματα που εξετάστηκαν.

Συνολικά μπορούμε να πούμε ότι το bogofilter κάτω από τις συνθήκες των μετρήσεων που πραγματοποιήθηκαν παρουσίασε την χειρότερη απόδοση μεταξύ των προγραμμάτων και δεν μπόρεσε να πετύχει σωστό διαχωρισμό των μηνυμάτων.

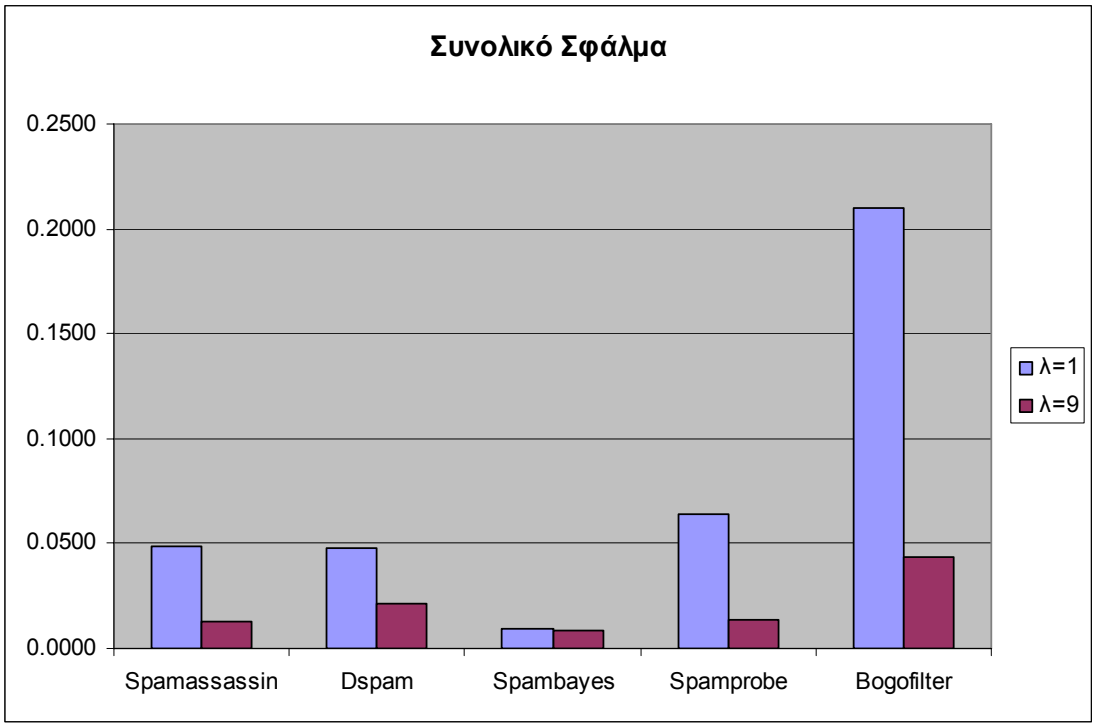
5.Σύνοψη αποτελεσμάτων

5.1 Μέτρηση 1





Στα παραπάνω διαγράμματα απεικονίζονται, για την μέτρηση 1, συνολικά τα σφάλματα False Positive και False Negative για όλα τα προγράμματα της δοκιμής. Μια βασική παρατήρηση που μπορούμε να κάνουμε είναι ότι τα περισσότερα προγράμματα εμφανίζουν αρκετά μεγαλύτερο σφάλμα False Negative από ότι False Positive. Αυτό κρίνεται μάλλον φυσιολογική και ορθή επιλογή των δημιουργών αυτών των προγραμμάτων αφού συνήθως η λάθος κατάταξη ενός επιθυμητού μηνύματος κοστίζει πολύ περισσότερο από την λάθος κατάταξη ενός ανεπιθύμητου. Ακραία συμπεριφορά στον τομέα αυτό εμφανίζει το πρόγραμμα Bogofilter του οποίου το σφάλμα False Negative είναι 0.44 ενώ το False Positive μόλις 0.0021. Δηλαδή το bogofilter κατατάσσει εσφαλμένα σχεδόν ένα στα δυο μηνύματα. Εξαίρεση σ'αυτή τη συμπεριφορά εμφανίζουν δυο από τα προγράμματα. Το Dspam εμφανίζει παρόμοια σφάλματα False Negative και False Positive ενώ το Spambayes εμφανίζει αρκετά μικρότερη τιμή σφάλματος False Negative.

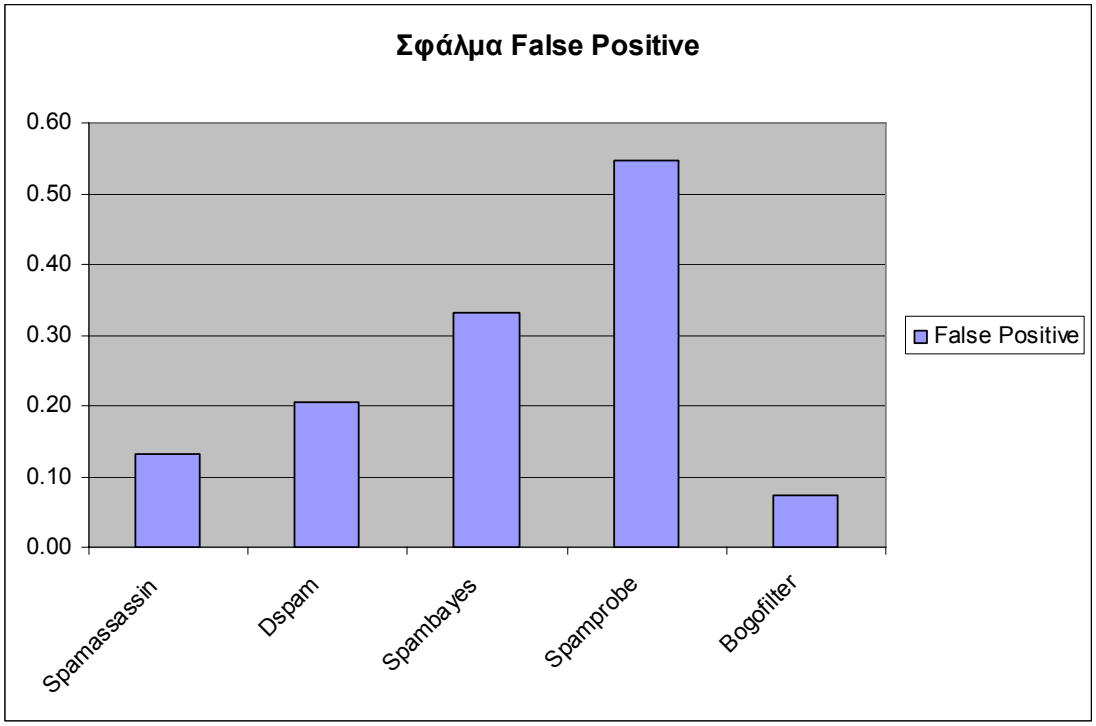


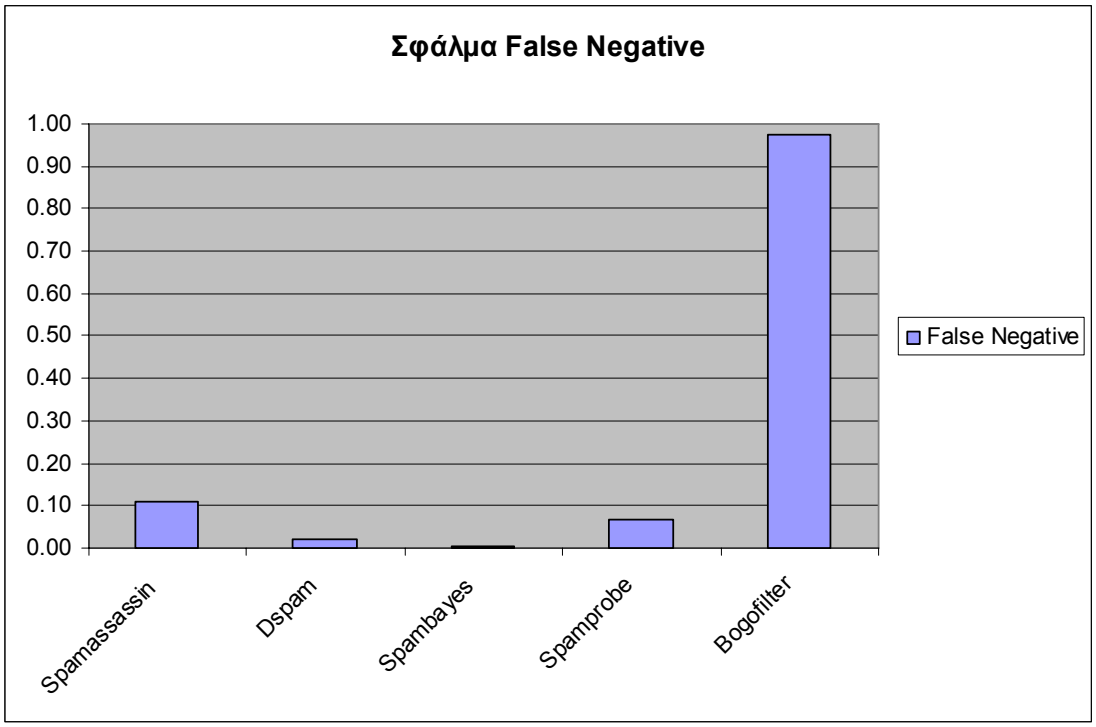
Όσον αφορά το συνολικό σφάλμα παρατηρούμε σχεδόν σε όλες τις εφαρμογές, πλην του Spambayes, αρκετά μικρότερη τιμή του σφάλματος για λ=9 από ότι για λ=1. Να θυμίσουμε ότι όταν λ=9 το σφάλμα False Positive θεωρείται ότι κοστίζει 9 φορές περισσότερο από το σφάλμα False Negative, κάτι που πλησιάζει αρκετά τις πραγματικές συνθήκες λειτουργίας των προγραμμάτων. Πράγματι στην πλειοψηφία

των χρηστών είναι πολύ σημαντικότερο να μην χάσουν κάποιο επιθυμητό μήνημα από το να λάβουν μερικά επιπλέον ανεπιθύμητα. Συνολικά λοιπόν μπορούμε να πούμε ότι την καλύτερη απόδοση στην πρώτη μέτρηση, με αρκετή διαφορά, παρουσιάζει το Spambayes το οποίο αποδίδει εξίσου καλά τόσο στον διαχωρισμό των ανεπιθύμητων όσο και στων επιθυμητών μηνημάτων. Στην δεύτερη θέση ακολουθεί το πρόγραμμα Spamassassin. Τα προγράμματα Dspam και Spamprobe εμφανίζουν μεγάλο σφάλμα False Positive και False Negative αντίστοιχα. Στην τελευταία θέση βρίσκεται το Bogofilter λόγω του πολύ μεγάλου σφάλματος False Negative που εμφανίζει.

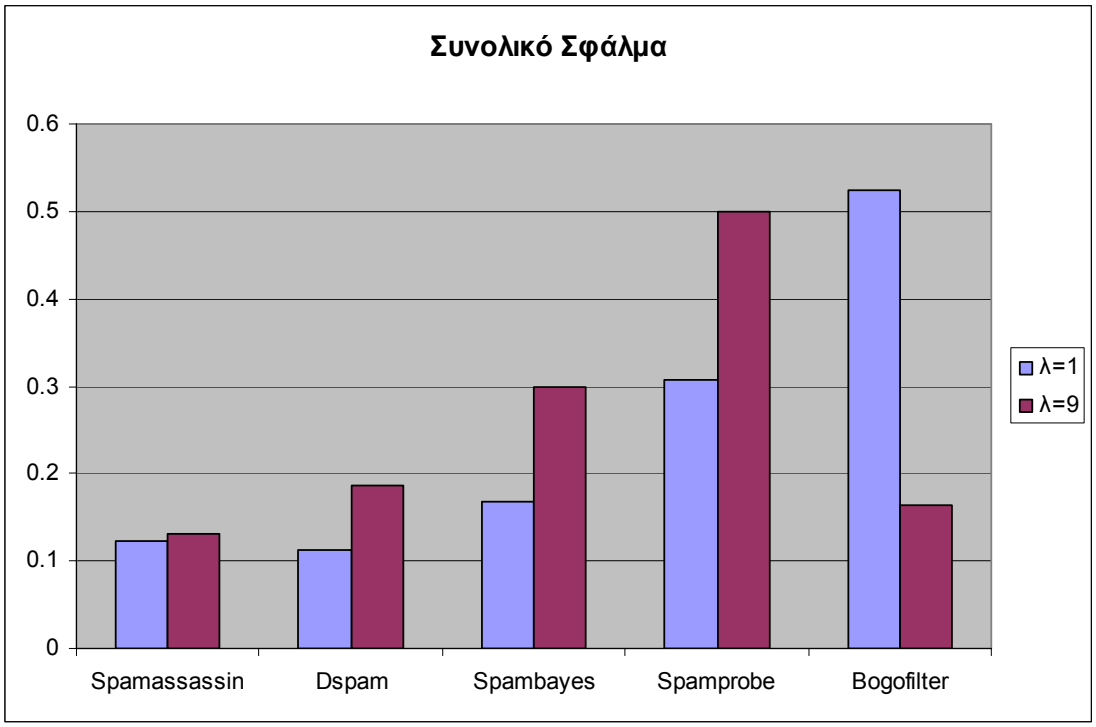
5.2 Μέτρηση 2

Η μέτρηση 2 εξετάζει την περίπτωση όπου εκπαίδευση γίνεται με ένα γενικό dataset από το site του spamassassin ενώ έλεγχος γίνεται με το dataset που δημιουργήθηκε με τα προσωπικά μηνύματα του γράφοντος.





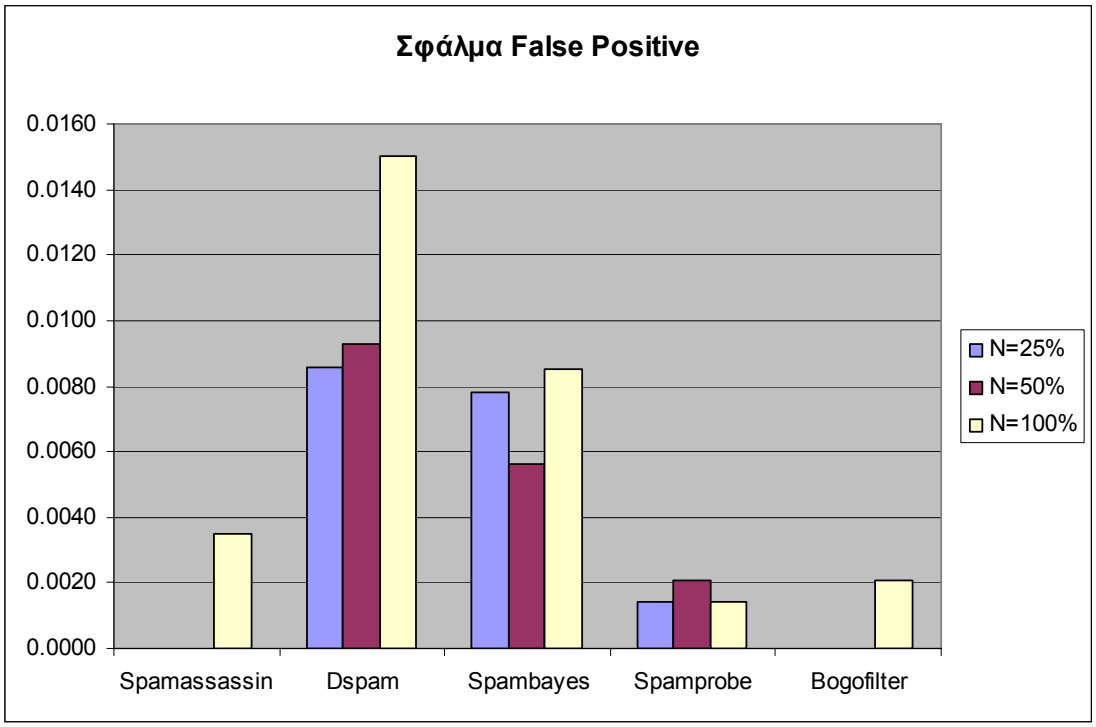
Στα παραπάνω διαγράμματα απεικονίζονται, για την μέτρηση 2, συνολικά τα σφάλματα False Positive και False Negative για όλα τα προγράμματα της δοκιμής. Αν εξαιρέσουμε το bogofilter, το οποίο μάλλον αποτυγχάνει να κάνει οποιονδήποτε διαχωρισμό στα ανεπιθύμητα μηνύματα, την καλύτερη απόδοση στον διαχωρισμό των επιθυμητών μηνυμάτων εμφανίζει το Spamassassin. Όσον αφορά τα ανεπιθύμητα μηνύματα, καλύτερο διαχωρισμό επιτυγχάνει το spambayes. Παρατηρούμε από τα διαγράμματα αυτά ότι τα περισσότερα από τα προγράμματα εμφανίζουν μεγάλη τιμή σφάλματος False Positive. Πράγματι φαίνεται ότι τα δεδομένα επιθυμητών μηνυμάτων που χρησιμοποιήθηκαν για εκπαίδευση δεν εμφανίζουν και τόσο μεγάλη συσχέτιση με τα δεδομένα που χρησημοποιήθηκαν για τον έλεγχο. Πρέπει λοιπόν να γίνεται εκπαίδευση με προσωπικά μηνύματα του χρήστη που θα χρησιμοποιεί κάποιο από τα προγράμματα αν θέλουμε να έχουμε την μέγιστη απόδοση.

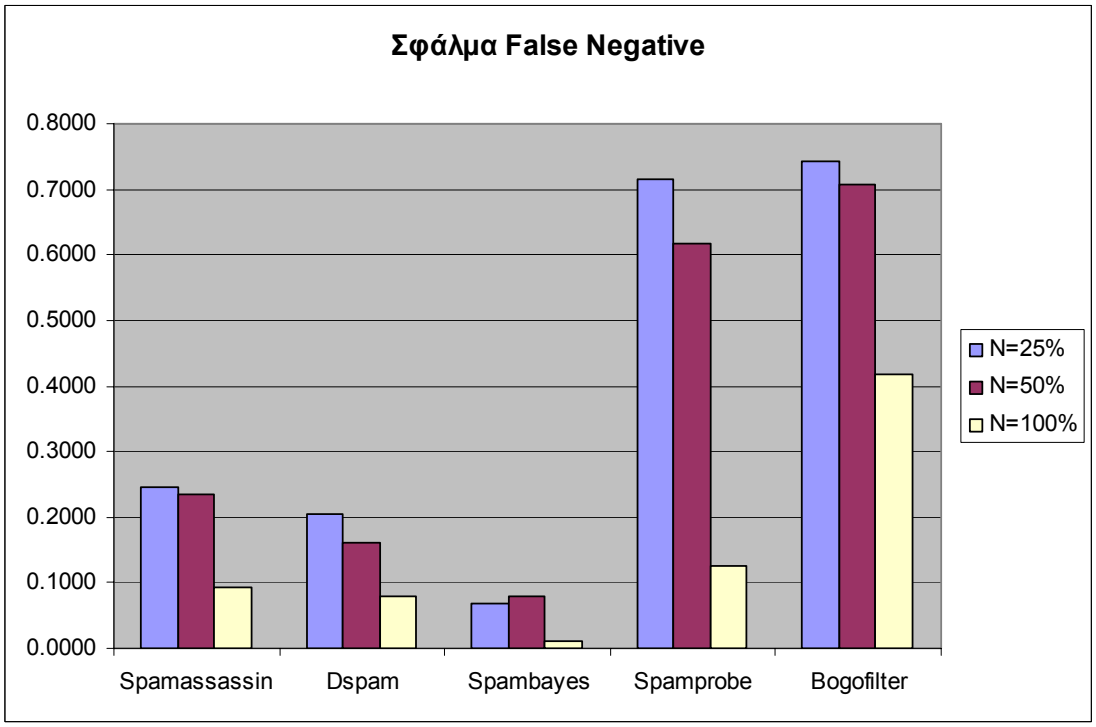


Από το διάγραμμα του συνολικού σφάλματος παρατηρούμε ότι για όλα σχεδόν τα προγράμματα, εκτός του Bogofilter του οποίου όμως η απόδοση δεν είναι αποδεκτή, το σφάλμα παίρνει μεγαλύτερη τιμή για λ=9 από ότι για λ=1 γεγονός που σημαίνει ότι για όλα τα προγράμματα το σφάλμα False Positive είναι μεγαλύτερο. Καλύτερη απόδοση εμφανίζει το Spamassassin και ακολουθούν τα Dspam, Spambayes και Spambrobe με την σειρά αυτή. Γενικά για όλα τα προγράμματα, εκτός ίσως του spamassassin, η απόδοση είναι αρκετά άσχημη και δεν θα ήταν σκόπιμη η χρήση τους σε περιβάλλον παραγωγής χωρίς εκπαίδευση με προσωπικά μηνύματα των χρηστών.

5.3 Μέτρηση 3

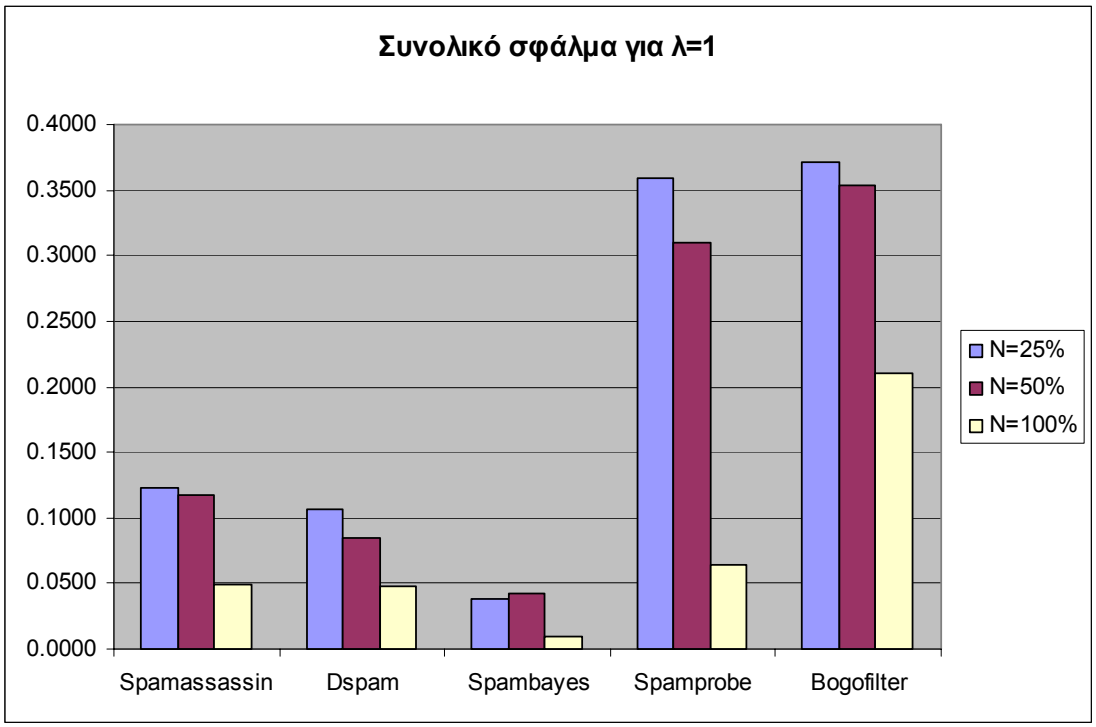
Στην μέτρηση 3 εξετάζουμε την μεταβολή της απόδοσης των προγραμμάτων αλλάζοντας το μέγεθος του dataset εκμάθησης. Έγιναν 3 δοκιμές για κάθε πρόγραμμα με μέγεθος 25%, 50% και 100% του dataset. Στην συνέχεια εξετάζουμε συγκεντρωτικά τα αποτελέσματα για όλα τα προγράμματα.

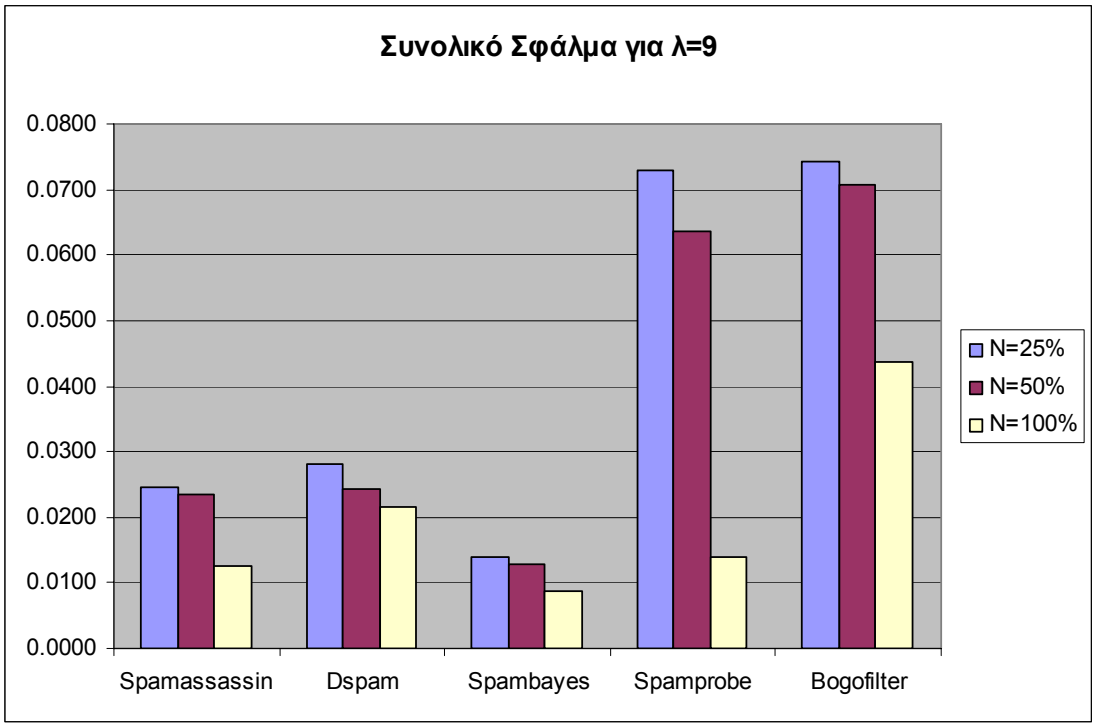




Στα δυο πρώτα διαγράμματα απεικονίζονται τα σφάλματα False Positive και False Negative για τα τρία μεγέθη dataset για όλα τα προγράμματα. Παρατηρούμε ότι το σφάλμα False Positive είτε παραμένει σταθερό είτε αυξάνεται αυξάνοντας το μέγεθος του dataset σε όλα τα προγράμματα που εξετάσθηκαν. Το σφάλμα False Negative

αντίθετα μειώνεται θεαμάτικά αυξάνοντας το μέγεθός του dataset για όλα τα προγράμματα. Αυτό μάλλον οφείλεται στο ότι οι κατασκευαστές των προγραμμάτων προσπάθησαν να προστατεύσουν τους χρήστες από τα σφάλματα False Positive για την περίπτωση που κάποιο τέτοιο πρόγραμμα τεθεί σε λειτουργία χωρίς να έχει πρώτα γίνει εκπαίδευση με κάποιο επαρκές μέγεθος dataset. Έτσι συνήθως τα προγράμματα προτιμούν να καταταξουν ένα μήνυμα ως επιθυμητό παρά ως ανεπιθύμητο όταν στην πραγματικότητα δεν έχουν επαρκείς πληροφορίες για να το κατατάξουν κανονικά. Η πιο εντυπωσιακή μείωση του σφάλματος False Negative με την αύξηση του πλήθους των μηνυμάτων του dataset παρατηρείται στο προγράμμα Spamprobe. Το σφάλμα παίρνει την τιμή 0.72 για dataset 25% ενώ για το dataset 100% η τιμή του είναι 0.12.





Στα δυο προηγούμενα διαγράμματα απεικονίζεται η μεταβολή του συνολικού σφάλματος, για τιμές λ=1 και λ=9, για όλα τα προγράμματα στα διάφορα μεγέθη dataset.

Εξετάζοντας το συνολικό σφάλμα βλέπουμε ότι όλα τα προγράμματα ανεξαιρέτως εμφανίζουν βελτίωση της απόδοσης τους καθώς αυξάνεται το μέγεθος του dataset. Ανάμεσά τους ξεχωρίζει σίγουρα το Spambayes το οποίο εμφανίζει ακόμη και για το μικρότερο dataset αισθητά καλύτερη απόδοση από τα άλλα προγράμματα. Ακολουθεί το Spamassassin που έχει μικρό προβάδισμα σε σχέση με το Dspam εξαιτίας της καλύτερης απόδοσης του στον διαχωρισμό των επιθυμητών μηνυμάτων. Το Spamprobe ξεκινά με άσχημη απόδοση η οποία όμως βελτιώνεται εντυπωσιακά καθώς αυξάνεται το μέγεθος του dataset. Τέλος χειρότερη απόδοση εμφανίζει το bogofilter το οποίο βέβαια βελτιώνει την απόδοσή του καθώς αυξάνεται το μέγεθος του dataset χωρίς όμως να μπορέσει να πλησιάσει την απόδοση των υπόλοιπων προγραμμάτων.

5.4 Συνολικά

Αν προσπαθούσαμε να εκφέρουμε μια άποψη για την συνολική απόδοση των προγραμμάτων λαμβάνοντας υπ' όψη τα αποτελέσματα των μετρήσεων που έγιναν

τότε μάλλον την πρώτη θέση μεταξύ τους θα έπαιρνε το spambayes. Το πρόγραμμα αυτό παρουσίασε, με αρκετή διαφορά από τα υπόλοιπα, την καλύτερη απόδοση στις μετρήσεις 1 και 3. Είναι λοιπόν αυτό που θα επιδείξει την καλύτερη συμπεριφορά σε πραγματικές συνθήκες λειτουργίας. Μοναδικό σημείο στο οποίο συνίσταται προσοχή είναι η εκπαίδευσή του με δεδομένα των μηνυμάτων που δέχονται οι χρήστες και όχι κάποιο γενικό dataset από το Internet, καθώς όπως φαίνεται από την μέτρηση 2 η απόδοσή του μειώνεται σε μεγάλο βαθμό όταν τα δεδομένα εκπαίδευσης δεν είναι αρκετά συσχετισμένα με τα δεδομένα ελέγχου. Αντίστοιχη μεγάλη μείωση της απόδοσης βεβαίως συναντούμε και στα υπόλοιπα προγράμματα.

Στην δεύτερη θέση έρχεται το Spamassassin καθώς παρουσιάζει την δεύτερη καλύτερη απόδοση στις μετρήσεις 1 και 3, μετά το Spambayes, και την καλύτερη απόδοση στην μέτρηση 2. Το Spamassassin είναι λοιπόν μια καλή επιλογή αφού εμφανίζει πολύ καλή, αλλά όχι άριστη, απόδοση σε όλους τους τομείς και δεν μειονεκτεί σε κανέναν.

Ακολουθούν τα Spamprobe, dspam τα οποία εμφανίζουν προβλήματα σε διαφορετικούς τομείς. Το spamprobe τα πήγε πολύ καλά στην μέτρηση με το πλήρες dataset αλλά χάνει μεγάλο μέρος της απόδοσης του όταν γίνεται εκπαίδευση με μικρό dataset. Επίσης άσχημη είναι η απόδοσή του και στην μέτρηση 2. Το σημαντικότερο μειονέκτημα του dspam είναι ότι εμφανίζει αρκετά μεγάλο σφάλμα False Positive κάτι που το κάνει όχι και τόσο ελκυστικό στα μάτια των περισσοτέρων χρηστών.

Τέλος την χειρότερη επίδοση από όλα τα προγράμματα σε όλες τις μετρήσεις εμφάνισε το Bogofilter του οποίου η χρήση δεν συνίσταται.

6. Βιβλιογραφία

1. Mehran Sahami, Susan Dumais, David Heckerman, Eric Horvitz. A Bayesian Approach to Filtering Junk E-Mail.
2. Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakkis, Constantine D. Spyropoulos and Panagiotis Stamatopoulos. Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach.
3. Andrew McCallum, Kamal Nigam. A Comparison of Event Models for Naïve Bayes Text Classification
4. Paul Graham. Stopping Spam. http://www.paulgraham.com/stopspam.html
5. Frank Smadja and Henry Tumblin. Automatic Spam Detection as a Text Classification Task.
6. Paul Graham. A Plan for Spam. http://www.paulgraham.com/spam.html
7. Spamassassin. http://spamassassin.apache.org/
8. Spambayes. http://spambayes.sourceforge.net/
9. Dspam. http://dspam.nuclearelephant.com/
10. Brian Burton. Spamprobe – Bayesian Spam Filtering Tweaks. http://spamprobe.sourceforge.net/paper.html
11. Bogofilter. http://bogofilter.sourceforge.net/
12. Distributed Checksum Clearinghouse. http://www.rhyolite.com/anti-spam/dcc/
13. Jon Praed. How Lawsuits Against Spammers Can Aid Spam-Filtering Technology.
14. Spamhaus. http://www.spamhaus.org/
15. Gary Robinson.A Statistical Approach to the Spam Problem.2003. http://www.linuxjournal.com/article/6467
16. I. Rish. An empirical study of the naive Bayes classifier.
17. William S. Yerazunis. 2003. Sparse Binary Polynomial Hashing and the CRM114 Discriminator .
18. B. Baesens, M. Egmont-Petersen, R. Castelo and J. Vanthienen. Learning Bayesian network classifiers for credit scoring using Markov Chain Monte Carlo search.
19. William W. Cohen. Learning Rules that Classify E-Mail.

20. Bart Massey Mick Thomure Raya Budrevich Scott Long. Learning Spam: Simple Techniques For Freely-Available Software.
21. T.A Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system.
22. Georgios Sakkis, Ion Androutsopoulos, Georios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos and Panagiotis Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail.
23. Gregory L. Wittel and S. Felix Wu. On Attacking Statistical Spam Filters.
24. David Madigan. Statistics and The War on Spam.
25. Karl-Michael Schneider. A Comparison of Event Models for Naive Bayes Anti-Spam E-Mail Filtering
26. Beerud Dilip Sheth. A Learning Approach to Personalized Information Filtering.
27. Jason D. M. Rennie and Tommi Jaakkola. Automatic Feature Induction for Text Classification
28. Michael Salib. 2002. MeatSlicer: Spam Classification with Naive Bayes and Smart Heuristics.
29. Cormac O'Brien and Carl Vogel. Spam Filters: Bayes vs. Chi-squared; Letters vs. Words

Παράρτημα Α – Κώδικας

A.1 Spamassassin

A.1.1 Training

```
#!/usr/bin/env perl

use strict;
use warnings;

# script to train spamassasin with various sets of spam and ham

# folder with spam relative to corpus dir
my $spam="spam_2";
# folder with spam relative to corpus dir
my $ham="ham";

# home directory where the other folders are in
my $home_dir="/home/ankal/thesis";
my $corpus_dir="$home_dir/corpus";
my $sa_learn_options="--siteconfigpath=$home_dir/etc/spamassassin -L";

my $spam_dir="$corpus_dir/$spam";
my $ham_dir="$corpus_dir/$ham";

my $ls_files_spam=`ls -1 $spam_dir`;
my $ls_files_ham=`ls -1 $ham_dir`;
#print $ls_files;

# first erase any existing Bayes database
`rm -rf $home_dir/var/spamassassin/*`;
```

```
my $result_text=`sa-learn --no-sync $sa_learn_options --spam $spam_dir`;
$result_text=~/^Learned from/ || die "error learning spam $spam_dir\n";
my @files_spam=split('\n',$ls_files_spam);
foreach my $file (@files_spam) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$spam_dir."/".$file;
        #my $result_text=`sa-learn --no-rebuild --spam $fname`;
        #$result_text=~/^Learned from/ || die "error learning spam $fname\n";
        print "spam $spam $file_num\n";
}


$result_text=`sa-learn $sa_learn_options --no-sync --ham $ham_dir`;
$result_text=~/^Learned from/ || die "error learning ham $ham_dir\n";
my @files_ham=split('\n',$ls_files_ham);
foreach my $file (@files_ham) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$ham_dir."/".$file;
        #my $result_text=`sa-learn --no-rebuild --ham $fname`;
        #$result_text=~/^Learned from/ || die "error learning ham $fname\n";
        print "ham $ham $file_num\n";
}
system("sa-learn $sa_learn_options --sync");
#system("chmod 644 /var/spamassassin/Bayes_*");
```

A.1.2 Testing

```
#!/usr/bin/env perl

use strict;

# script to test performance for different corpuses
```

```
# v 1.0 the basics for spamassassin

# the corpus name
my $corpus="ankal_ham";
# the home directory
my $home_dir="/home/ankal/thesis";

my $corpus_dir="$home_dir/corpus/$corpus";

my $result_dir="$home_dir/results/spamassassin";

my $result_file="$result_dir/$corpus";

(-e $result_file) && die "$result_file already exists";

open FILE,">$result_file";

my $ls_files=`ls -1 $corpus_dir`;
#print $ls_files;

my @files=split('\n',$ls_files);

foreach my $file (@files) {
        my $file_num;
        if ($file=~/^(.*)\./) {
                $file_num=$1;
        } elsif ($file=~/^\d{5}/) {
                $file_num=$file;
        } else {
                die "wrong file name $file\n";
        }
        my $fname=$corpus_dir."/".$file;
        my $result_text = `cat $fname | spamc`;
```

```
	$result_text=~/BAYES_(\d\d)/ || ( (print "$corpus $file_num no Bayes score\n") && (next) );
	my $Bayes_result=$1;
	#$result_text=~/X-Spam-Status: (Yes|No), hits=(.*?) required/ || die "no spamassassin score\n";
	#my $spamassassin_result=$2;
	print "$corpus $file_num $Bayes_result\n";
	print FILE "$corpus $file_num $Bayes_result\n";
}
```

A.2 Dspam

A.2.1 Training

```
#!/usr/bin/env perl

use strict;
use warnings;

# script to train dspam

# folder with spam relative to corpus dir
my $spam="spam_2";
# folder with spam relative to corpus dir
my $ham="ham";

# home directory where the other folders are in
my $home_dir="/home/ankal/thesis";
my $corpus_dir="$home_dir/corpus";

my $spam_dir="$corpus_dir/$spam";
my $ham_dir="$corpus_dir/$ham";

my $ls_files_spam=`ls -1 $spam_dir`;
my $ls_files_ham=`ls -1 $ham_dir`;
```

```
#print $ls_files;

# first erase any existing Bayes database
`mysql -u root -e 'delete from dspam_preferences; delete from dspam_signature_data;
delete from dspam_stats ; delete from dspam_token_data' dspam`;

my @files_spam=split('\n',$ls_files_spam);
foreach my $file (@files_spam) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$spam_dir."/".$file;
        system("dspam_corpus -addspam root $fname") && die "Error learning
$fname";
        print "spam $spam $file_num\n";
}


my @files_ham=split('\n',$ls_files_ham);
foreach my $file (@files_ham) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$ham_dir."/".$file;
   system("dspam_corpus root $fname") && die "Error learning $fname";
        print "ham $ham $file_num\n";
}
```

A.2.2 Testing

```
#!/usr/bin/env perl

use strict;

# v 1.0 for dspam

# the corpus name
```

```
my $corpus="ankal_ham";
# the home directory
my $home_dir="/home/ankal/thesis";

my $corpus_dir="$home_dir/corpus/$corpus";

my $result_dir="$home_dir/results/dspam";

my $result_file="$result_dir/$corpus";

(-e $result_file) && die "$result_file already exists";

open FILE,">$result_file";

my $ls_files=`ls -1 $corpus_dir`;
#print $ls_files;

my @files=split('\n',$ls_files);

foreach my $file (@files) {
        my $file_num;
        if ($file=~/^(.*)\./) {
                $file_num=$1;
        } elsif ($file=~/^\d{5}/) {
                $file_num=$file;
        } else {
                die "wrong file name $file\n";
        }
        my $fname=$corpus_dir."/".$file;
        my $result_text = `cat $fname | dspam --client --user root --classify`;
        $result_text=~/^X-DSPAM-Result: root; result="(.*?)"; probability=(.*?);
confidence=(.*)$/ || ( (print "$corpus $file_num no Bayes score\n") && (next) );
        my $Bayes_result=$2;
   my $Bayes_confidence=$3;
```

```
        #$result_text=~/X-Spam-Status: (Yes|No), hits=(.*?) required/ || die "no spamassassin score\n";
        #my $spamassassin_result=$2;
        print "$corpus $file_num $Bayes_result $Bayes_confidence\n";
        print FILE "$corpus $file_num $Bayes_result $Bayes_confidence\n";
}
```

A.3 Spambayes

A.3.1 Training

```
#!/usr/bin/env perl

use strict;
use warnings;

# script to train spambayes

# folder with spam relative to corpus dir
my $spam="spam_2";
# folder with spam relative to corpus dir
my $ham="ham";

# home directory where the other folders are in
my $home_dir="/home/ankal/thesis";
my $corpus_dir="$home_dir/corpus";

my $spam_dir="$corpus_dir/$spam";
my $ham_dir="$corpus_dir/$ham";

my $Bayes_db_file="$home_dir/var/spambayes/spambayesdb";

#print $ls_files;
```

```
# first erase any existing Bayes database
if (-e $Bayes_db_file) {
   `rm -f $Bayes_db_file`;
   `sb_filter.py -n`;
}

system("sb_mboxtrain.py -s $spam_dir -f");
system("sb_mboxtrain.py -g $ham_dir -f");
```

A.3.2 Testing

```
#!/usr/bin/env perl

use strict;

# v 1.0 for spambayes

# the corpus name
my $corpus="ham_2";
# the home directory
my $home_dir="/home/ankal/thesis";

my $corpus_dir="$home_dir/corpus/$corpus";

my $result_dir="$home_dir/results/spambayes";

my $result_file="$result_dir/$corpus";

(-e $result_file) && die "$result_file already exists";

open FILE,">$result_file";

my $ls_files=`ls -1 $corpus_dir`;
#print $ls_files;
```

```
my @files=split('\n',$ls_files);

foreach my $file (@files) {
	my $file_num;
	if ($file=~/^(.*)\./) {
		$file_num=$1;
	} elsif ($file=~/^\d{5}/) {
		$file_num=$file;
	} else {
		die "wrong file name $file\n";
	}
	my $fname=$corpus_dir."/".$file;
	my $result_text = `cat $fname | sb_filter.py`;
	$result_text=~/X-Spambayes-Classification: (.*?); (.\...)/ || ( (print "$corpus $file_num no Bayes score\n") && (next) );
	my $Bayes_result=$2;
	#$result_text=~/X-Spam-Status: (Yes|No), hits=(.*?) required/ || die "no spamassassin score\n";
	#my $spamassassin_result=$2;
	print "$corpus $file_num $Bayes_result\n";
	print FILE "$corpus $file_num $Bayes_result\n";
}
```

A.4 Spamprobe

A.4.1 Training

```
#!/usr/bin/env perl

use strict;
use warnings;

# script to train spamprobe
```

```
# folder with spam relative to corpus dir
my $spam="spam_2";
# folder with spam relative to corpus dir
my $ham="ham";

# home directory where the other folders are in
my $home_dir="/home/ankal/thesis";
my $corpus_dir="$home_dir/corpus";

my $spam_dir="$corpus_dir/$spam";
my $ham_dir="$corpus_dir/$ham";

my $db_dir="$home_dir/var/spamprobe";

my $ls_files_spam=`ls -1 $spam_dir`;
my $ls_files_ham=`ls -1 $ham_dir`;
#print $ls_files;

# first erase any existing Bayes database
(-e "$db_dir/sp_words") && `rm -f $db_dir/*`;


my @files_spam=split('\n',$ls_files_spam);
foreach my $file (@files_spam) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$spam_dir."/".$file;
        system("spamprobe -d $db_dir spam $fname") && die "Error learning
$fname";
        print "spam $spam $file_num\n";
}


my @files_ham=split('\n',$ls_files_ham);
```

```
foreach my $file (@files_ham) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$ham_dir."/".$file;
   system("spamprobe -d $db_dir good $fname") && die "Error learning $fname";
        print "ham $ham $file_num\n";
}
```

A.4.2 Testing

```
#!/usr/bin/env perl

use strict;

# v 1.0 for spamprobe

# the corpus name
my $corpus="ankal_ham";
# the home directory
my $home_dir="/home/ankal/thesis";

my $corpus_dir="$home_dir/corpus/$corpus";

my $result_dir="$home_dir/results/spamprobe";

my $db_dir="$home_dir/var/spamprobe";

my $result_file="$result_dir/$corpus";

(-e $result_file) && die "$result_file already exists";

open FILE,">$result_file";

my $ls_files=`ls -1 $corpus_dir`;
#print $ls_files;
```

```
my @files=split('\n',$ls_files);

foreach my $file (@files) {
	my $file_num;
	if ($file=~/^(.*)\./) {
		$file_num=$1;
	} elsif ($file=~/^\d{5}/) {
		$file_num=$file;
	} else {
		die "wrong file name $file\n";
	}
	my $fname=$corpus_dir."/".$file;
	my $result_text = `cat $fname | spamprobe -d $db_dir score`;
	$result_text=~/^(.*?) (.*?) (.*?)$/ || ( (print "$corpus $file_num no Bayes score\n") && (next) );
	my $Bayes_result=$2;
	#$result_text=~/X-Spam-Status: (Yes|No), hits=(.*?) required/ || die "no spamassassin score\n";
	#my $spamassassin_result=$2;
	print "$corpus $file_num $Bayes_result\n";
	print FILE "$corpus $file_num $Bayes_result\n";
}
```

A.5 Bogofilter

A.5.1 Training

```
#!/usr/bin/env perl

use strict;
use warnings;

# script to train bogofilter
```

```
# folder with spam relative to corpus dir
my $spam="spam_2";
# folder with spam relative to corpus dir
my $ham="ham";

# home directory where the other folders are in
my $home_dir="/home/ankal/thesis";
my $corpus_dir="$home_dir/corpus";

my $spam_dir="$corpus_dir/$spam";
my $ham_dir="$corpus_dir/$ham";

my $config_file="$home_dir/etc/bogofilter/bogofilter.cf";

my $ls_files_spam=`ls -1 $spam_dir`;
my $ls_files_ham=`ls -1 $ham_dir`;
#print $ls_files;

# first erase any existing Bayes database
(-e "/home/ankal/thesis/var/bogofilter/wordlist.db" ) && (`rm -f $home_dir/var/bogofilter/wordlist.db`);

my @files_spam=split('\n',$ls_files_spam);
foreach my $file (@files_spam) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$spam_dir."/".$file;
        system("cat $fname | bogofilter -c $config_file -s") && die "Error learning $fname";
        print "spam $spam $file_num\n";
}
```

```
my @files_ham=split('\n',$ls_files_ham);
foreach my $file (@files_ham) {
        $file=~/^(.*)\./ || die "wrong filename\n";
        my $file_num=$1;
        my $fname=$ham_dir."/".$file;
   system("cat $fname | bogofilter -c $config_file -n") && die "Error learning $fname";
        print "ham $ham $file_num\n";
}
```

A.5.2 Testing

```
#!/usr/bin/env perl

use strict;

# v 1.0 for bogofilter

# the corpus name
my $corpus="ankal_ham";
# the home directory
my $home_dir="/home/ankal/thesis";

my $corpus_dir="$home_dir/corpus/$corpus";

my $result_dir="$home_dir/results/bogofilter";

my $result_file="$result_dir/$corpus";

my $config_file="$home_dir/etc/bogofilter/bogofilter.cf";

(-e $result_file) && die "$result_file already exists";

open FILE,">$result_file";
```

```
my $ls_files=`ls -1 $corpus_dir`;
#print $ls_files;

my @files=split('\n',$ls_files);

foreach my $file (@files) {
	my $file_num;
	if ($file=~/^(.*)\./) {
		$file_num=$1;
	} elsif ($file=~/^\d{5}/) {
		$file_num=$file;
	} else {
		die "wrong file name $file\n";
	}
	my $fname=$corpus_dir."/".$file;
	my $result_text = `cat $fname | bogofilter -c $config_file -p`;
	$result_text=~/X-Bogosity: (.*?), tests=(.*?), spamicity=(.*?), version=/ || (
(print "$corpus $file_num no Bayes score\n") && (next) );
	my $Bayes_result=$3;
	print "$corpus $file_num $Bayes_result\n";
	print FILE "$corpus $file_num $Bayes_result\n";
}
```

A.6 Πρόγραμμα Διαμόρφωσης Αποτελεσμάτων

```
#!/usr/bin/perl

use strict;
use warnings;

# record count
my $rc=0;
# initialize scores
my %scores;
```

```
    $scores{0}=0;
    $scores{5}=0;
    $scores{10}=0;
    $scores{15}=0;
    $scores{20}=0;
    $scores{25}=0;
    $scores{30}=0;
    $scores{35}=0;
    $scores{40}=0;
    $scores{45}=0;
    $scores{50}=0;
    $scores{55}=0;
    $scores{60}=0;
    $scores{65}=0;
    $scores{70}=0;
    $scores{75}=0;
    $scores{80}=0;
    $scores{85}=0;
    $scores{90}=0;
    $scores{95}=0;
    $scores{99}=0;

while (<>) {
    my @fields=split " ";
    chomp @fields;
    $rc++;
    # convert decimal to 1-100
    if ($fields[2]=~/\./) {
       my $t=sprintf("%.2f", $fields[2]);
       $fields[2]=$t*100;
    }
    SWITCH: {
       if ($fields[2]>=0 && $fields[2]<=1) { $scores{0}++ ; last SWITCH;}
       if ($fields[2]>1 && $fields[2]<=5) { $scores{5}++ ; last SWITCH;}
```

```
        if ($fields[2]>5 && $fields[2]<=10) { $scores{10}++ ; last SWITCH;}
        if ($fields[2]>10 && $fields[2]<=15) { $scores{15}++ ; last SWITCH;}
        if ($fields[2]>15 && $fields[2]<=20) { $scores{20}++ ; last SWITCH;}
        if ($fields[2]>20 && $fields[2]<=25) { $scores{25}++ ; last SWITCH;}
        if ($fields[2]>25 && $fields[2]<=30) { $scores{30}++ ; last SWITCH;}
        if ($fields[2]>30 && $fields[2]<=35) { $scores{35}++ ; last SWITCH;}
        if ($fields[2]>35 && $fields[2]<=40) { $scores{40}++ ; last SWITCH;}
        if ($fields[2]>40 && $fields[2]<=45) { $scores{45}++ ; last SWITCH;}
        if ($fields[2]>45 && $fields[2]<55) { $scores{50}++ ; last SWITCH;}
        if ($fields[2]>=55 && $fields[2]<60) { $scores{55}++ ; last SWITCH;}
        if ($fields[2]>=60 && $fields[2]<65) { $scores{60}++ ; last SWITCH;}
        if ($fields[2]>=65 && $fields[2]<70) { $scores{65}++ ; last SWITCH;}
        if ($fields[2]>=70 && $fields[2]<75) { $scores{70}++ ; last SWITCH;}
        if ($fields[2]>=75 && $fields[2]<80) { $scores{75}++ ; last SWITCH;}
        if ($fields[2]>=80 && $fields[2]<85) { $scores{80}++ ; last SWITCH;}
        if ($fields[2]>=85 && $fields[2]<90) { $scores{85}++ ; last SWITCH;}
        if ($fields[2]>=90 && $fields[2]<95) { $scores{90}++ ; last SWITCH;}
        if ($fields[2]>=95 && $fields[2]<99) { $scores{95}++ ; last SWITCH;}
        if ($fields[2]>=99 && $fields[2]<=100) { $scores{99}++ ; last SWITCH;}
        die "Error score is $fields[2]\n";
    }
    #print "$fields[0]-$fields[1]-$fields[2]\n"; # debug

}
my %r;
$r{0}=$scores{0}*100/$rc; printf("0 %.2f\n",$r{0});
$r{5}=$scores{5}*100/$rc; printf ("5 %.2f\n",$r{5});
$r{10}=$scores{10}*100/$rc; printf ("10 %.2f\n",$r{10});
$r{15}=$scores{15}*100/$rc; printf ("15 %.2f\n",$r{15});
$r{20}=$scores{20}*100/$rc; printf ("20 %.2f\n",$r{20});
$r{25}=$scores{25}*100/$rc; printf ("25 %.2f\n",$r{25});
$r{30}=$scores{30}*100/$rc; printf ("30 %.2f\n",$r{30});
$r{35}=$scores{35}*100/$rc; printf ("35 %.2f\n",$r{35});
$r{40}=$scores{40}*100/$rc; printf ("40 %.2f\n",$r{40});
```

```
$r{45}=$scores{45}*100/$rc; printf ("45 %.2f\n",$r{45});
$r{50}=$scores{50}*100/$rc; printf ("50 %.2f\n",$r{50});
$r{55}=$scores{55}*100/$rc; printf ("55 %.2f\n",$r{55});
$r{60}=$scores{60}*100/$rc; printf ("60 %.2f\n",$r{60});
$r{65}=$scores{65}*100/$rc; printf ("65 %.2f\n",$r{65});
$r{70}=$scores{70}*100/$rc; printf ("70 %.2f\n",$r{70});
$r{75}=$scores{75}*100/$rc; printf ("75 %.2f\n",$r{75});
$r{80}=$scores{80}*100/$rc; printf ("80 %.2f\n",$r{80});
$r{85}=$scores{85}*100/$rc; printf ("85 %.2f\n",$r{85});
$r{90}=$scores{90}*100/$rc; printf ("90 %.2f\n",$r{90});
$r{95}=$scores{95}*100/$rc; printf ("95 %.2f\n",$r{95});
$r{99}=$scores{99}*100/$rc; printf ("99 %.2f\n",$r{99});

#print "$rc\n"; #debug
```