Bayes Argorithm

Υλοποίηση 1 αλγορίθμου αφελή ταξινομητή Bayes με την παραδοχή ότι οι τιμές/λέξεις είναι ανεξάρτητες μεταξύ τους.
Για να μπορέσουν να βγουν τα αποτελέσματα σωστά χρησιμοποιήθηκε και η εκτιμήτρια Laplace η οποία πρόσθεση (+) μια ψεύτο εμφάνιση σε όλες τις λέξεις.
Για την υλοποιήση του αλγορίθμου χρησιμοποιήθηκαν τα εξής:

- Ένα HashMap ονόματι MailHash.

  > Ο MailHash περιέχει για όλο το λεξιλόγιο τον αριθμό τον mails τα οποία περιέχουν την συγκεκριμένη λέξη.

- Ένα HashSet ονόματι tempSet.

  > Στο οποίο αποθηκεύονται προσωρινά όλες οι μοναδικές λέξεις για κάθε mail. Και έπειτα προσθέτονται στο MailHash(Ένα HashMap)

Διάβασμα στοιχείων

Για να επιτευχθεί αυτό χρησιμοποιούνται δύο συναρτήσεις: inputToHashMap() και UpdateMailHash()

- inputToHashMap

  > Παίρνει δύο ορίσματα το path το οποίο αναφέρεται σε ποια δεδομένα θα επεξεργαστεί ο αλγόριθμος και το posost που αναφέρεται για πόσο της % τον δεδομένων θα εκπαιδευτεί.Η inputToHashMap() για κάθε directory ανοίγει ένα mail το καταγράφει αν είναι spam η ham και το εισάγει στο τοπικό tempSet, και τέλος ενημερώνει τον λεξιλόγιο δηλαδή τον MailHash για της νέες η τις υπάρχουσες λέξεις αντίστοιχα. *UpdateMailHash Παίρνει ένα όρισμα 0 για spam και 1 για ham και ελέγχει αν η λέξη υπάρχει η όχι και αναλόγως αυξάνει τον κατάλληλο μετρητή.

Μετά το διάβασμα τον δεδομένων καλείται η συνάρτηση.

Training

- Laplace()

> η οποία προσθέτει (+) μια ψευτό εμφάνιση σε όλες τις λέξεις.

- Probability()

> η οποία αλλάζει την τιμή μέσα στον MailHash(Ένα HashMap) και το κάνει πιθανότητα εμφάνιση της λέξεις σύμφωνα με τον τύπο

$$SpamPropability_{lexi} = log(1 + lexi^{\varphi o \rho \acute{\epsilon} \varsigma \epsilon \mu \varphi \acute{\alpha}} / SpamCounter^{\sigma \upsilon \nu o \lambda \iota \kappa o \varsigma})$$

$$HamPropability_{lexi} = log(1 + lexi^{\varphi o \rho \acute{\epsilon} \varsigma \epsilon \mu \varphi \acute{\alpha}} / HamCounter^{\sigma \upsilon \nu o \lambda \iota \kappa o \varsigma})$$

Testing

Κατά την διάρκεια του testing ο αλγόριθμος διαβάζει το τελευταίο κομμάτι των δεδομένων που είναι άγνωστα μέχρι στιγμής στον αλγόριθμο και προσπαθεί με βάση τα trainig δεδομένα να κάνει μία πρόβλεψη σε ποια κατηγορία ανήκει μέσω των 2 παρακάτω τύπων ανάλογα με πιο έχει την μεγαλύτερη πιθανότητα.

- εάν η λέξη υπάρχει στο mail.

$$P(C = 1/X) = log(P(C = 1)) + log(\prod_{i=1}^{forallwords} P(X_i = x_i/C = 1))$$

$$P(C = 0/X) = log(P(C = 0)) + log(\prod_{i=1}^{forallwords} P(X_i = x_i/C = 0))$$

- εάν η λέξη δεν υπάρχει στο mail.
- $$P(C = 1/X) = log(P(C = 1)) + log(1/TotalHamCounter))$$
- $$P(C = 0/X) = log(P(C = 0)) + log(1/TotalSpamCounter))$$

Υπολογισμός (Accuracy PRecision Recall)

- Accuracy

$$\frac{100 * (TP + TN)}{TP + TN + FP + FN}$$

- HamPRecision

$$\frac{100 * TP}{TP + FP}$$

- SpamPRecision

$$\frac{100 * TN}{TN + FN}$$

- RecallHam

$$\frac{100 * TP}{TP + FN}$$

- RecallSpam

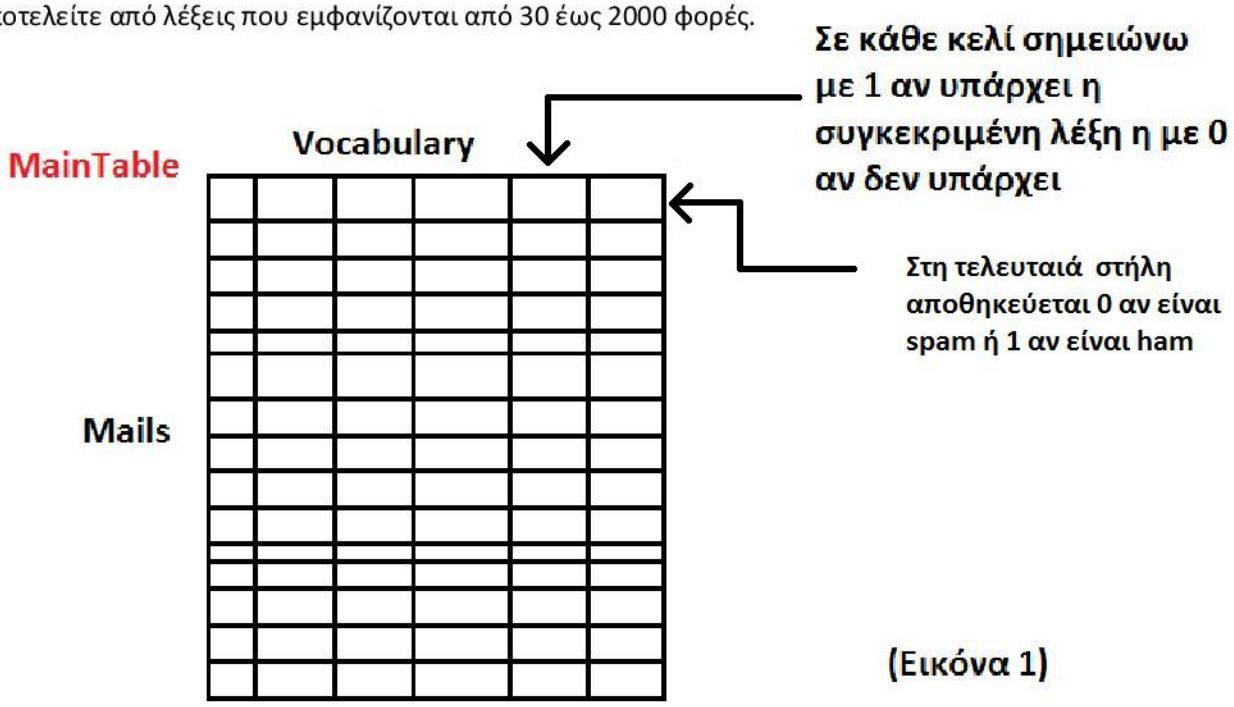
$$\frac{100 * TN}{TN + FP}$$

TP=0,TN=1,FP=2,FN=3

Id3

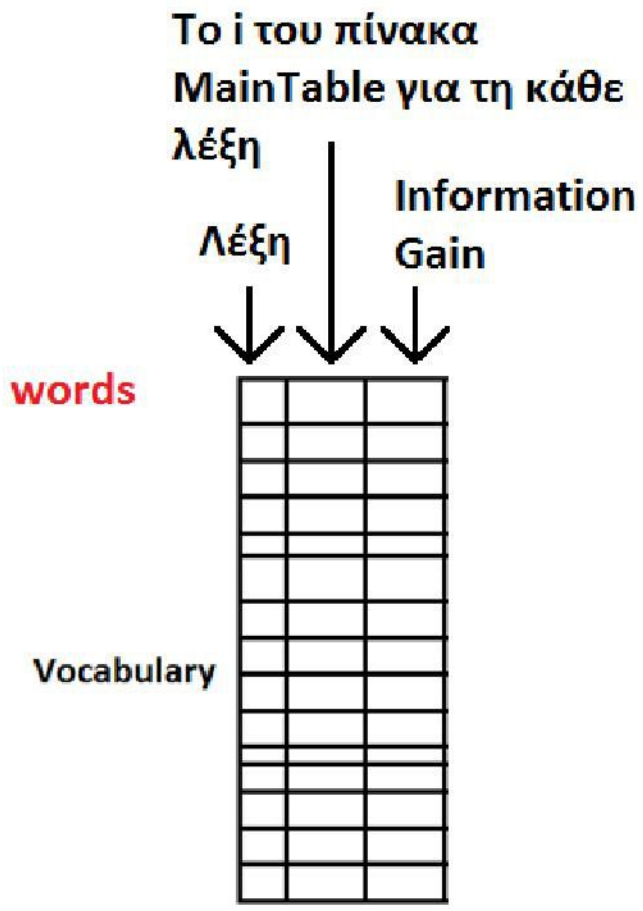
Ο αλγόριθμος ID3 κατασκευάζει δέντρα από τα παραδείγματα εκπαίδευσης. Σε κάθε εσωτερικό κόμβο ελέγχουμε την τιμή μιας ιδιότητας.(Υπάρχει η δεν υπάρχει μια συγκεκριμένη λέξη). Σε κάθε κόμβο του δένδρου μετράμε τα παραδείγματα κάθε κατηγορίας και επιλέγουμε μέσω ευρετικής την ιδιότητα που τα διαχωρίζει καλύτερα. Χρησιμοποιεί ως ευρετική συνάρτηση αξιολόγησης των ιδιοτήτων το κέρδος πληροφορίας (IG). Οι πιθανότητες εκτιμώνται τώρα από τα παραδείγματα αυτού του κόμβου. Σταματάμε όταν η πιθανότητα να είναι spam ή η πιθανότητα να είναι legit είναι μεγαλύτερη του 75%.

Για την υλοποίηση του Id3 φτιάξαμε δυο κλάσεις. Η πρώτη ονομάζεται PrepereID3Data η οποία δέχεται τα δεδομένα εκπαίδευσης και τα αποθηκεύει στην ακόλουθη μορφή(Εικόνα 1) με τη μέθοδο Read. Στη συνέχεια με τη μέθοδο Pruning δημιουργούμε το vocabulary μας ανάλογα με τον αριθμό εμφάνισης των λέξεων. Πιο συγκεκριμένα το vocabulary μας αποτελείτε από λέξεις που εμφανίζονται από 30 έως 2000 φορές.



(Εικόνα 1)

Έπειτα με τη βοήθεια της μεθόδου HashWordToTable δημιουργούμε το πίνακα words ο οποίος έχει την ακόλουθη μορφή(εικόνα 2) με σκοπό να συσχετίσουμε της στήλες του MainTable με κάθε λέξη.



(Εικόνα 2)

Διαδικασία εύρεσης αποτελέσματος: Μετά από την εισαγωγή δεδομένων εκπαίδευσης κατασκευάζουμε την κλάση Id3Data η οποία δέχεται τα test δεδομένα τα αναπαριστά στην μορφή Εικόνα 1 και αφού έχουμε υπολογίσει το IG κάθε λέξης επιλέγουμε τη καλύτερη και βλέπουμε αν υπάρχει η αν δεν υπάρχει στο test mail. Σε περίπτωση που υπάρχει κρατάμε τα δεδομένα εκπαίδευσης που υπάρχει αυτή η λέξη αλλιώς κρατάμε τα δεδομένα στα οποία δεν υπάρχει αυτή η λέξη. Κατόπιν υπολογίζουμε το IG κάθε λέξης από τα δεδομένα εκπαίδευσης που έχουν μείνει και συνερίζουμε την ιδιά διαδικασία έως ότου το 75% των δεδομένων εκπαίδευσής να είναι είτε spam είτε legit.

Για την παραπάνω διαδικασία χρησιμοποιήθηκαν οι εξής μέθοδοι:

•AddIgToTable(table)=Η μέθοδος αυτή υπολογίζει κάθε φορά το IG για κάθε λέξη από τα δεδομένα εκπαίδευσης που έχουν απομείνει και ενημερώνει τη τρίτη στήλη του πίνακα words.

•SortIgTable()=Η μέθοδός αυτή ταξινομεί το πίνακα words με βάση το IG και επιστρέφει τη λέξη με το μεγαλύτερο IG.

•ID3Result()= Η μέθοδος αυτή δέχεται τα test δεδομένα τα αναπαριστά στην μορφή εικόνα 1 και στη συνέχεια εκτελεί τη διαδικασία εύρεσης αποτελέσματος.

Logistic Regression

Αρχικά, με την InputToHashMap( ) δημιουργούμε το Vocabulary το οποίο αποτελέιται από όλες μοναδικές λέξεις που περιέχονται στα training mails. Στη συνέχεια, μειώνουμε το πλήθος λέξεων του Vocabulary κρατώντας όσες λέξεις εμφανίζονται απο 30 μέχρι 2000 φορές, με την μέθοδο Prunning( ).

Αφού δημιουργήσουμε το Vocabulary,με την μέθοδο Read( ) φτιάχνουμε τον πίνακα MainTable[ ], ο οποίος έχει ως γραμμές τα training mails και ως στήλες το vocabulary. Αν στο mail (i) περιέχεται η λέξη (j) τότε βάζουμε 1 στο κατάλληλο κελί. Στην τελευταία στήλη του MainTable[ ] αποθηκεύεται αν το συγκεκριμένο μήνυμα είνα spam ή ham.

Παράλληλα , φτιάχνουμε και τον δυσδιάστατο πίνακα words[ ], του οποίου οι γραμμές είναι οι ιδιότητες-λέξεις. Στην πρώτη στήλη αποθηκεύται η λέξη, στη δεύτερη στήλη αποθηκεύεται θέση στην οποία βρίσκεται στον MainTable[] και στην τρίτη το βάρος της κάθε ιδιότητας-λέξης.

Η μέθοδος regression( ), αποτελείται από μια for( ) η οποία τρέχει τόσες επαναλήψεις όσες η μεταβλητή epoxes. Ακόμα, μια εμφωλευμένη for( ) η οποία τρέχει για όλες τις λέξεις του πίνακα words και ενημερώνει για κάθε λέξη το αντίστοιχο βάρος της στην τρίτη στήλη του πίνακα. Κατά την ενημέρωση των βαρών καλείται η μέθοδος MailClassify( ) η οποία υπολογίζει την f(x)=w*x για όλες τις ιδιότητες με τα αντίστοιχα βάρη του πίνακα words[ ]. Η MailClassify() καλεί την μέθοδο sigmoid() στην οποία περνάει ως όρισμα την f(x) που έχει υπολογίσει και η sigmoid με τη σειρά της επιστρέφει το απότελεσμα της σιγμοειδούς σύναρτησης με όρισμα την f(x).

Αφού έχει υπολογίσει τα βάρη για κάθε ιδιότητα, με την μέθοδο calculateTest(), παίρνει ένα καινούργιο mail από τα test δεδομένα, καλεί την MailClassify() η οποία επιστρέφει μέσω της sigmoid() μια πιθανότητα. Αν η τιμή αυτής της πιθανότητας είναι μεγαλύτερη απο την 1-πιθανότητα, τότε το mail είναι ham, στην αντίθετη περίπτωση είναι spam.