Μιχαλακάκου Κωνσταντίνα-P3150111
2019-2020
Πληροφορική,ΟΠΑ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ-Εργασία $2_{η}$

Γενικά:

Το πρόγραμμα αποτελείται από δύο java αρχεία τα Test.java,Train.java

Main: Test.java

Arguments:

args[0]:path/to/folder/pui

args[1]:πλήθος φακέλων που θα χρησιμοποιηθούν για training απ'τά 10.

Πχ C:\Users\User\Desktop\pu1 9

Train.java

ReadFile:

Έχει ορίσματα τον φάκελο pui και το πλήθος των testing φακέλων και Διαβάζει τα μηνύματα που περιέχονται στους training φακελους και μετράει τον αριθμό εμφανίσεως κάθε λέξης στα spams και στα hams αντιστοιχα.

Συγκεκριμένα:

Διαβάζουμε ένα ένα τα αρχεία μέσα στους φακέλους και για κάθε λέξη εάν είναι σε Spam if(bool) και δεν έχει εμφανιστεί ξανά στο ίδιο txt !multOccuranceS.containsKey(words[j]) προσθέτουμε +1 στον αριθμό εμφανίσεων της στην spam. Αλλιώς αν είναι σε ham και εμφανίζεται πρώτη φορά στο txt προσθέτουμε +1 στην ham.

Props:

Έχει ορίσματα τον φάκελο pui και το πλήθος των testing φακέλων. Υπολογίζει την πιθανότητα κάθε λέξης με δεδομένο ότι το μήνυμα είναι ham και spam:P(wordi|ham) και P(wordi|spam)

Εφαρμόζει εξομάλυνση laplace για αποφυγή μηδένισης γινομένου και log για αποφυγή underflow.

Μιχαλακάκου Κωνσταντίνα-P3150111
2019-2020
Πληροφορική,ΟΠΑ

Συγκεκριμένα:

Αφού καλέσουμε την ReadFile για να αρχικοποιηθούν τα hashmaps ham και spam υπολογίζουμε για κάθε λέξη στην spam τον log της πιθανότητας P(wordi|spam)

:*log((double)(spam.get(i)+1)/(spam.size()+countWords))*

Όπου
spam.get(i)/spam.size():
οι εμφανίσεις της λέξης στα spams / το σύνολο των διαφορετικών λέξεων στα spams

Ταυτόχρονα +1 στον αριθμητη ενώ +countWords το σύνολο των διαφορετικών λέξεων στον παρανομαστή για την εξομάλυνση Laplace.

Με τον ίδιο τρόπο υπολογίζουμε και τα P(wordi|ham).

Εάν η λέξη δεν υπάρχει καθόλου στα hams ορίζουμε την συχνότητα της λέξης 0 στον υπολογισμό της log(P(wordi|ham))

```
if (!ham.containsKey(i)){

        WordHam = log((double)(0+1)/(ham.size()+countWords));
```

ενώ αντίχτοιχα αν υπάρχει στην ham και όχι στην spam ορίζουμε την συχνότητα της λέξης 0 στον υπολογισμό της log(P(wordi|spam)).

Έτσι τα WordHamProps και WordSamProps περιέχουν ως κλειδιά όλες τις λέξεις που εμφανίζονται στα μηνύματα και ως τιμή τις πιθανότητες που περιγράφηκαν παραπάνω για τα hams και τα props αντίστοιχα.

Test:

Η test είναι η main μας η οποία διαβάζει κάθε ένα testing φάκελο(10-training files) και για κάθε λέξη που βρίσκει αθροίζει τα log των πιθανοτήτων τους όπως υπολογίστηκαν στην Train ξεχωριστά για Spam και Ham :totalS,totalH.

Έπειτα στο άθροισμα totalS προσθέτει τον log του πλήθους των λέξεων των Spams και στο totalH των Hams ώστε να προκύψει ο αριθμητής για την δεσμευμένη πιθανότητα

Μιχαλακάκου Κωνσταντίνα-P3150111
2019-2020
Πληροφορική,ΟΠΑ

log(P(spam|xi))=**log(P(spam)*P(xi|spam)/**P(xi)) και
log(P(ham|xi))=**log(P(ham)*P(xi|ham)/**P(xi)): probS,probH

Αφού συγκρίνει τα δύο αυτά log probS>probH αυξάνει το πλήθος των spams(spamsFound++) ή hams που βρήκε το πρόγραμμα καθώς και το πλήθος των spams που ήταν πράγματι spams (tp++) ,και το πλήθος ham που ήταν πράγματι ham (tn++).

Έπειτα υπολογίζει την τιμή της ανάκλησης ,της ακρίβειας,του F1 και του accuracy:Recall,Percision,f1,accuracy.

Πειράματα:

Τρέχοντας το πρόγραμμα για τον φάκελο PU1 και για διαφορετικό πλήθος δεδομένων εκπαίδευσης κάθε φορά έχουμε τον παρακάτω πίνακα .

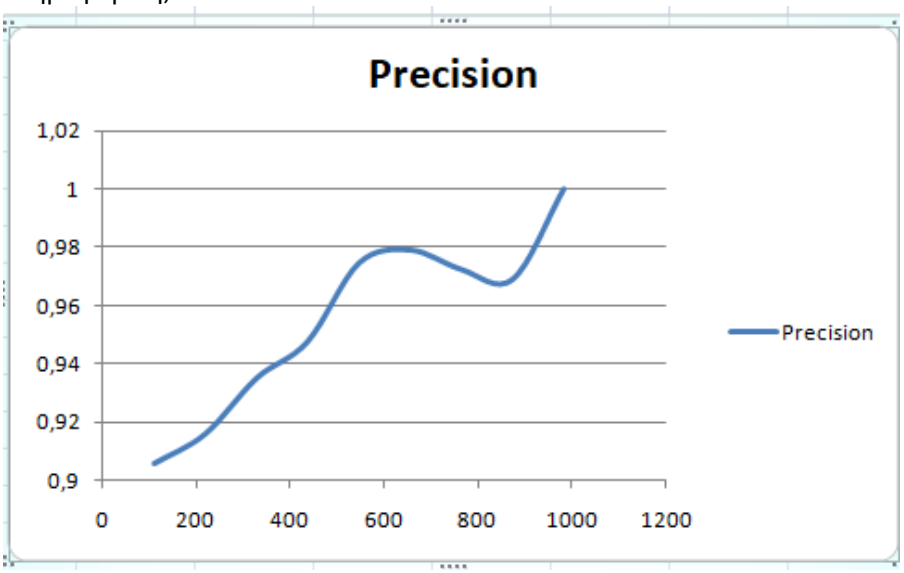
Η μεταβλητή X αναφέρεται στο πλήθος των διαφορετικών λέξεων που χρησιμοποιήθηκαν για training ανάλογα με το πλήθος των parts τ ορίσαμε ως είσοδο.

X	Precision	Recall	f1	Accuracy
109	0,9058	0,9351	0,9202	0,9286
218	0,916	0,9375	0,9266	0,9346
327	0,9351	0,9434	0,9392	0,9462
436	0,9477	0,9444	0,946	0,9525
545	0,9746	0,9625	0,9685	0,9724
654	0,9791	0,9791	0,9791	0,9816
763	0,9724	0,9791	0,9757	0,9785
872	0,969	0,9791	0,974	0,977
981	1	0,9791	0,9894	0,9908

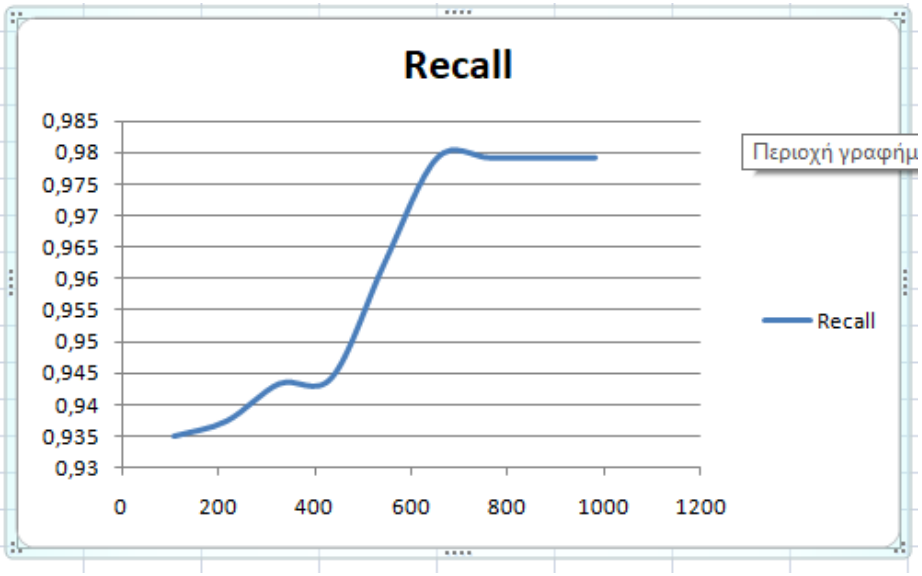
Οι καμπύλες που προκύπτουν απτις παραπάνω τιμές είναι οι εξείς:

Precision:

Μιχαλακάκου Κωνσταντίνα-Ρ3150111
2019-2020
Πληροφορική,ΟΠΑ

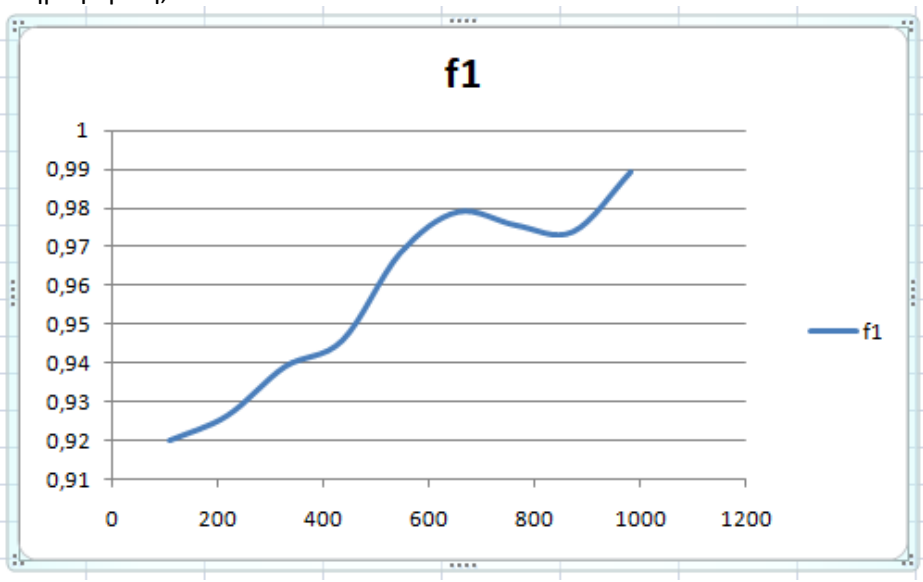


Recall:



F1:

Μιχαλακάκου Κωνσταντίνα-P3150111
2019-2020
Πληροφορική,ΟΠΑ



Accuracy: