ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ ΜΕ C++

(4η Εργασία)

ΜΠΟΓΔΑΝΟΣ ΜΙΧΑΗΛ (3100123)

ΜΑΓΚΟΣ ΡΑΦΑΗΛ-ΓΕΩΡΓΙΟΣ(3100098)

Αρχείο main.cpp

Ο ρόλος του αρχείου αυτού είναι ελέγχει τις παραμέτρους που δίνει ο χρήστης στην γραμμή εντολών για την εγκυρότητά τους.Αυτό επιτυγχάνεται με τις παρακάτω μεθόδους:

- bool isNumeric(const char* str) Συνάρτηση που ελέγχει το αλφαριθμητικό όρισμα χαρακτήρα-χαρακτήρα και επιστρέφει true μόνο όλοι οι χαρακτήρες είναι αριθμοί.
- bool fexists(const char *filename) Συνάρτηση που ελέγχει εάν το αλφαριθμητικό όρισμα(που παριστάνει όνομα αρχείου) υπάρχει.Το ifstream επιστρέφει 1 σε περίπτωση που το αρχείο υπάρχει,0 αλλιώς.
- void checkArgsNum(int&length) throw(IllegalNumberOfArgumentsException)Ελέγχει την εγκυρότητα του αριθμό ορισμάτων(αναφορά length-ψευδώνυμο argc).Ο αριθμός πρέπει να είναι είτε 5 (χωρίς όρισμα –scores και με default threshold),είτε 6(είτε με –scores ενεργοποιημένο είτε με τιμή του threshold),είτε 7 (ενεργοποιημένο –scores και τιμή του threshold).Εάν ο αριθμός είναι άκυρος,προκύπτει εξαίρεση τύπου IllegalNumberOfArgumentsException (ορισμός στο αρχείο exception.h παρακάτω).
- unsigned defineThreshold(int len,bool scores,char *arg[]) Επιστρέφει την τιμή που θα ανατεθεί στην μεταβλητή threshold.Το όρισμα len είναι το argc,το scores καθορίζει εάν το –scores ενεργοποιήθηκε ή όχι και τέλος το όρισμα argv της main(πίνακας δεικτών σε ακεραίους).Η επιστρεφόμενη τιμή προκύπτει με την λογική ότι εάν το –scores ενεργοποιήθηκε και ο αριθμός ορισμάτων είναι 7 (συμπεριλαμβανομένου και του argv[0],δηλ.το όνομα του προγράμματος) ελέγχουμε το τελευταίο όρισμα(θέση του threshold).Εάν είναι αριθμητικό τότε επιστρέφουμε την τιμή του με χρήση της atoi(μετατροπή από πίνακα χαρακτήρων σε unsigned),αλλιώς τυπώνεται προειδοποίηση ότι η τιμή του threshold θα γίνει default(10).Εάν το –scores είναι ανενεργό και ο αριθμός ορισμάτων είναι 6 και το όρισμα είναι αριθμητικό ομοίως με χρήση atoi επιστρέφουμε την αριθμητική τιμή.Αλλιώς τυπώνεται πάλι προειδοποίηση.Τέλος εάν φτάσαμε στο τέλος της συνάρτησης και δεν έχει επιστρέψει,τότε είναι ώρα να επιστρέψουμε την default τιμή 10.
- void checkValidityOfArgs(int &num, char *args[], int startpos) throw(IllegalArgumentsException) Συνάρτηση που ελέγχει την εγκυρότητα των αρχείων που δίνονται ως ορίσματα στην γραμμή εντολών.Αυτά τα ορίσματα θα βρίσκονται είτε στις θέσεις 4 και 5 του πίνακα ορισμάτων ,εάν το –scores είναι ενεργό και στις θέσεις 5 και 6 εάν το –scores ανενεργό.Τα αρχεία που θα δοθούν ξανά από τον χρήστη(μέσω της συνάρτησης void reEnterFile(char* s)) θα ελεγχθούν και πάλι για την εγκυρότητά τους.
- int main(int argc, char* argv[]) Η συνάρτηση main ουσιαστικά αξιοποιεί τις παραπάνω συναρτήσεις,χειρίζεται τις εξαιρέσεις καθώς και αναθέτει τις τιμές των μεταβλητών hamKeyword spamKeywords.Ακολούθως υπάρχει αυτούσιο το κομμάτι που βρίσκεται στην εκφώνηση της εργασίας.

Αρχείο exception.h

Το αρχείο αυτό περιέχει τον ορισμό της εξαίρεσης IllegalNumberOfArgumentsException που προκύπτει σε περίπτωση που το πλήθος τον ορισμάτων είναι μικρότερο ή μεγαλύτερο του απαιτούμενου.Ο κατασκευαστής παίρνει ως όρισμα ένα string που είναι το μήνυμα του σφάλματος που θα τυπωθεί στο cerr και ως δεύτερο όρισμα το πλήθος των ορισμάτων που λανθασμένα εισήγαγε ο χρήστης.

Αρχείο wordscore.h

Το αρχείο αυτό περιέχει τον ορισμό της τάξης WordScore όπως υπάρχει στην εκφώνηση.

Αρχείο featureselector.h

Το αρχείο αυτό περιέχει τον ορισμό της τάξης FeatureSelector.

Private μέλη της τάξης FeatureSelector :

- const char* hamFileNames : το αρχείο που περιέχει τα ονόματα των ham .
- const char* spamFileNames: το αρχείο που περιέχει τα ονόματα των spam.
- const unsigned threshold: το όρισμα που δίνεται από τον χρήστη στην main.

Protected μέλη της τάξης FeatureSelector :

vector<WordScore> spam: Περιέχει αντικείμενα WordScore (λέξεις που βρέθηκαν στην συλλογή spam) που πληρούν τις προυποθέσεις του threshold.

vector<WordScore> ham: Περιέχει αντικείμενα WordScore (λέξεις που βρέθηκαν στην συλλογή ham) που πληρούν τις προυποθέσεις του threshold.

Public μέλη της τάξης FeatureSelector :

FeatureSelector(char* hfn, char* sfn, unsigned th) :hamFileNames(hfn), spamFileNames(sfn), threshold(th) Κατασκευαστής όπου δίνουμε στα ορίσματα τις τιμές τους.Επίσης καλεί την μέθοδο star που ξεκινά την επεξεργασία.

- static unsigned *spamsize :* πληθος μηνυμάτων spam.Αρχικοποιείται με 0.
- static unsigned *hamsize :* πλήθος μηνυμάτων hamΑρχικοποιείται με 0.

- const const_iterator hamBegin() : Επιστρέφει έναν const_iterator στην αρχή του vector ham.
- const const_iterator hamEnd() : Επιστρέφει έναν const_iterator στο τέλος του vector ham.
- const const_iterator spamBegin() : Επιστρέφει έναν const_iterator στην αρχή του vector spam.
- const const_iterator spamEnd() : Επιστρέφει έναν const_iterator στο τέλος του vector spam.
- double calcHamFm(unsigned sw, unsigned hw) : Υπολογίζει το spamFmeasure.Ελέγχει το όρισμα sw ώστε αν είναι 0,επιστρέφει 0 γιατί σε διαφορετική περίπτωση προκύπτει NaN.
- double calcSpamFm(unsigned sw, unsigned hw) : Υπολογίζει το hamFmeasure.Ελέγχει το όρισμα hw ώστε αν είναι 0,επιστρέφει 0 γιατί σε διαφορετική περίπτωση προκύπτει NaN.
- void start() : Η μέθοδος αυτή είναι υπέυθυνη για την επεξεργασία που ζητείται.Γίνεται χρήση 2 map<string,unsigned> ένα για spam και ένα για ham.Ακολούθως διαβάζονται τα αρχεία ham και με την χρήση του ham map βρίσκονται οι συχνότητες εμφάνισης κάθε λέξης.Ομοίως ακολούθως γίνεται η ίδια διαδικασία για τα spam.Εάν κάποιο αρχείο δεν βρεθεί,τυπώνεται μήνυμα λάθους.Στη συνέχεια,με χρήση map const_iterator ελέγχουμε αν κάθε λέξη του spam map υπάρχει και στον ham map.Εάν ναι,τότε υπολογίζεται με χρήση των μεθόδων calcHamFm/calcSpamFm και τοποθετείται στο κατάλληλο vector το αντικείμενο WordScore.Τα προηγούμενα γίνονται όμως μόνο εάν ικανοποιείται η συνθήκη του threshold.Ακολούθως διαγράφεται το στοιχείο από τον ham map,αφού η επεξεργασία του ολοκληρώθηκε εδώ.Εάν δεν βρέθηκε το στοιχείο και στον ham map,τότε τοποθετούμε το αντικείμενο WordScore στον spam vector αφού υπολογίσουμε το spamFmeasure(το hamFmeasure δεν ορίζεται).Στη συνέχεια,διασχίζουμε τον ham map και τοποθετούμε τα αντικείμενα αφού υπολογίσουμε πρώτα το hamFmeasure τους.Ακολούθως ταξινομούμε τα vector που προέκυψαν με την sort της βιβλιοθήκης algorithm(ο τελεστής < έχει οριστεί στην κλάση WordScore ώστε να μπορεί να γίνει η απαραίτητη σύγκριση).
- Εσωτερική τάξη const_iterator :

Private μέλη :

- vector<WordScore> *current : δείκτης σε vector<WordScore> (θα είναι είτε ο ham vector είτε ο spam)
- vector<WordScore> *other : δείκτης σε vector<WordScore> (θα είναι ο εναπομένων vector)
- unsigned index : το τρέχον σημείο στον current vector.
- bool vspam : μεταβλητή bool ώστε να γνωρίζουμε εάν ο current είναι ο spam(true) ή ο ham(false).
- double findScore(string w) : επιστρέφει την τιμή του Fmeasure του αντικειμένου WordScore στον vector other που έχει τιμή word ίση με w.

Public μέλη :

- const_iterator(vector<WordScore> *pt, vector<WordScore> *ot, unsigned i,bool b):current(pt), other(ot), index(i), vspam(b) Κατασκευαστής που αναθέτει τις κατάλληλες τιμές στα ορίσματα.

- bool operator!=(const const_iterator& right) : Συγκρίνει την μεταβλητή index του ορίσματος με την μεταβλητή index της τρέχουσας τάξης και επιστρέφει το αποτέλεσμα της σύγκρισης μεταξύ τους με το διάφορο.
- const WordScore operator++(int) : Αυξάνει το index και επιστρέφει το αντικείμενο στη νέα θέση του vector.
- void print(ostream &out, bool b) : Συνάρτηση που τυπώνει τα αποτελέσματα.Εάν είναι ενεργοποιημένο το –scores (όρισμα b) τότε μαζί με την λέξη τυπώνονται και τα spamFmeasure και hamFmeasure της.Στη θέση index ανακτάμε το αντικείμενο WordScore και τυπώνουμε την μεταβλητή word (χρήση getWord μεθόδου) καθώς και την score(getScore μέθοδος).Επειδή όμως κάθε αντικείμενο περιλαμβάνει μόνο είτε το spamFmeasure είτε το hamFmeasure του,κάνουμε αναζήτηση στο vector που δείχνει το other κάνοντας χρήση της findScore.Σε περίπτωση που δεν είναι ενεργοποιημένο το scores,τυπώνουμε μόνο το word.