Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα
Εργασία:3η

Μέλη της ομάδας:
Βουρτζούμη Ουρανία Α.Μ:115201600024 Κοσμάς Αλέξανδρος Α.Μ:1115201700299
Βαρώνος Διονύσης Α.Μ:1115201600017

UniTesting:

Για το uni testing χρησιμοποιούμε την βιβλιοθήκη acutest.h
Με την εντολή make test δημιουργείται το εκτελέσιμο test το οποίο εκτελείται με την εντολή make test-run. Με την εντολή make test-clean διαγράφονται διαγράφεται το εκτελέσιμο test και όσα .o αρχεία δημιουργήθηκαν. Το εκτελέσιμο δημιουργείται μέσα στον φάκελο test αλλά όλες οι παραπάνω εντολές δίνονται στον αρχικό φάκελο.

Μεταγλώττιση κώδικα:

Η υλοποίηση μας για το δημιουργεί ένα εκτελέσιμο sigmod το οποίο δημιουργείται με την εντολή make. Με την εντολή make clean διαγράφεται το εκτελέσιμο ολα τα .o αρχεία και όλα τα .csv αρχεία τα οποία έχουν δημιουργηθεί κατά την εκτέλεση του προγράμματος.

Εντολή εκτέλεσης προγράμματος :

Το εκτελέσιμο εκτελείται με την εντολή:

make run ARGS="- d Directory -w datacsv"

ή απλώς

./sigmod - d Directory -w datacsv

Όπου:

Directory:

ο φάκελος που έχει τους φακέλους με τα .json αρχεία δηλαδή ο φάκελος 2013_camera_specs στην συγκεκριμένη άσκηση (ο φάκελος δεν υπάρχει στον φάκελο μας στο github αλλά θα πρέπει να υπάρχει στο directory που τρέχει το εκτελέσιμο sigmod)

ΣΗΜΑΝΤΙΚΌ: ως directory θα πρέπει να δωθει το όνομα του φακέλου και όχι ο φάκελος σαν path . Δηλαδή το σωστό είναι 2013_camera_specs και όχι 2013_camera_specs/

datacsv:

το όνομα του csv αρχείου που περιέχει τα ταιριασματα των καμερων(πχ sigmod_medium_labelled_dataset.csv ή sigmod_large_labelled_dataset.csv)

Output προγράμματος:

Το πρόγραμμα κατά την εκτέλεση του δημιουργεί ένα νέο αρχείο με όνομα Same.csv που περιέχει όλες τις θετικές συσχετισεις που δημιουργήθηκαν απο τις κλικες μέσω του 60% των δεδομένων του dataset W και ένα αρχείο με όνομα Different.csv με τις αντίστοιχες αρνητικές συσχετισεις.

Στην συνέχεια εκτυπώνει το Success rate του μοντέλου κατά την διαδικασία του testing το οποίο γίνεται με το επόμενο 20% του datasetw.

Κατά την διάρκεια του προγράμματος εκτυπώνωνται κάποιες προτάσεις που μας ενημερώνουν πως το πρόγραμμα μόλις τελείωσε κάποια συγκεκριμένη διαδικασία (πχ όταν διαβάσει όλα τα δεδομένα του datasetX)

Ροή προγράμματος :

Το πρόγραμμα δημιουργεί δημιουργεί μια δομή Hash στην οποία αποθηκευονται τα στοιχεία του κάθε json αρχείου απο το datasetX και μια δομή LHash η οποία αντιπροσοπευει το vocabulary όλων των json.

Για κάθε αρχείο json αποθηκεύεται το id της κάμερα μέσα στην δομή Hash και στην συνέχεια διαβάζουμε το json λέξη - λέξη όπου κάθε λέξη αποθηκεύεται στο λεξιλόγιο και σε μια δομή WHash που αποθηκεύεται στην αντίστοιχη θέση που αποθηκευτικε το id του json μέσα στην δομή Hash.

Για κάθε νέα λέξη που βρίσκει το λεξιλόγιο την αποθηκεύει και για κάθε λέξη που έχει ήδη αυξάνει κατά ένα την μεταβλητή wordperj που αντιστοιχεί στην συγκεκριμένα λέξη και αντιπροσοπευει τον αριθμό των json στα οποία βρέθηκε αυτή η λέξη.

Αντίστοιχα η δομή WHash αποθηκεύει κάθε νέα λέξη του json και για κάθε ήδη υπάρχουσα αυξάνει κατά ένα την μεταβλητή τάδε που αντιστοιχεί στο πόσες φορές βρέθηκε η λέξη αυτή σε αυτό το.

Μόλις διαβαστουν όλες οι λέξεις του json το WHash υπολογίζει το tf της κάθε λέξης του και στην συνέχεια για κάθε μία απ της λέξης του δίνει το tf στο λεξιλόγιο το οποίο το προσθέτει στην μεταβλητή tfcount της αντίστοιχης λέξης και με αυτόν τον τρόπο κρατάει το άθροισμα των tf της κάθε λέξης για όλα τα json.

Αφού τελειώσει η παραπάνω διαδικασία για όλα τα αρχεία του datasetX η δομή LHash υπολογίζει το μέσο tf-idf της κάθε λέξεις και

το αποθηκεύει στην μεταβλητή isf και αποθηκεύει το idf της λέξης στην μεταβλητή tfcount.

Στην συνέχεια το λεξιλόγιο ταξινομει τις λέξεις με βάση το μεγαλύτερο μέσο tf-idf και κρατάει τις 1000 πρώτες.

Στην συνέχεια για κάθε στοιχείο της δομής Hash μια δομη Hvector η οποία είναι ένας πίνακας κατακερματισμου που αντιπροσοπευει ενα spar array.
Δηλαδή για κάθε μια λέξη απο τις 1000 σημαντικές του λεξηλογίου εαν υπάρχει στην κάμερα αποθηκεύεται η θέση της και η τιμή tf*idf της αντίστοιχης λέξης για την συγκεκριμένη κάμερα.
Με αυτόν τον τρόπο δεν αποθηκεύουμε στο vector μας τις τιμές που ισουνται με 0 και είναι περιττή πληροφορία.

Μολις δημιουργείται το κάθε vector διαγράφεται η δομή WHash αυτής της θέσης.

Στην συνέχεια δημιουργούνται οι κλικες βάση το 60% των θετικών και το 60% των αρνητικών συσχετήσεω του dataW και οι οι αρνητικές συσχετισεις μεταξύ των κλικων που δεν τεριαζουν.

Επίσης δημιουργούνται τα αρχεία Testing.csv και Validation.csv που το κάθε ένα περιέχει 20% αρνητικων και θετικών συσχετησεων και θα χριεαστουν στην συνέχεια για να πάρουμε τα δεδομένα στις αντίστοιχες διαδικασίες.

Αφού δημιουργηθουν οι κλικες και οι αρνητικές συσχετισεις το πρόγραμμα δημιουργεί τα αρχεία Same.csv και Different.csv που σε αυτά αποθηκευονται όλες οι θετικές συσχετισεις βάση των κλικων της δομής και όλες οι αρνητικές συσχετισεις αντίστοιχα.

Στην συνέχεια μεσω της συνάρτησης make input δημιουργούμε το input το οποίο θα δοθεί στο μοντέλο για την διαδικασία της εκπαίδευσης (δηλαδή το training set) .
Το input δημιουργείται από το περιεχόμενο των αρχείων Different και Same.csv.
Πιο συγκεκριμένα η δομή input έχει ένα πίνακα που κρατάει το concatenation κάθε ζευγαριού και ένα πίνακα που για καθε ζευγάρι κρατάει 0 αν ειναι αρνιτική και 1 αν ειναι θετεκή η συσχέτιση.

Αφού δημιουργηθεί το input πάμε στην διαδικασία της εκπαίδευσης το input πάμε στην διαδικασία της εκπαίδευσης στην διαδικασία της εκπαίδευσης που είναι ως εξής:

Μέσα σε μία επανάληψη γίνονται τα εξής βήματα:

Αρχικά γίνεται το training του μοντέλου μέσω της συνάρτησης.

Η συνάρτηση training εκπαιδεύει το μοντέλο σε δέσμες των 1000 παρατηρήσεων(batch size)

Κάθε φορά δίνονται τόσες δέσμες όσες είναι ο αριθμός των threads που μπορεί να εκτελέσει ο Job Scheduler.

Ο Job Scheduler εκτελεί οποιαδήποτε ρουτίνα η οποία δίνεται σε μορφη δομης job.
Στην συγκεκριμένη περίπτωση η ρουτίνα που εκτελίται είναι η Job training που λειτουργέι ως εξής:

Για κάθε παρατήρηση υπολογίζει την τιμή του μοντέλου και την απόκλιση του μοντέλου για αυτήν την πρόβλεψη.
Στην συνέχεια για κάθε θέση του concatenation υπολογίζουμε το γινόμενο της τιμής της συγκεκριμένης θέσης επι της παραπάνω τιμής. Αυτό το γινόμενο επι της τιμής (0,6) το αφαιρούμε κάθε φορά απο το βάρος της αντίστοιχης θέσης.

Η διαδικασία αυτή επαναλαμβάνεται μέχρι να δοθούν όλες οι παρατηρήσεις δηλαδή όλα τα ζευγάρια του input.

Αφού τελειώσει η διαδικασία της εκπαίδευσης στην συνέχεια για όσα στοιχεία του datasetX δεν ανήκουν σε κάποια κλίκα γίνεται η μεταξύ τους πρόβλεψη από το μοντέλο.

Αν η πρόβλεψη είναι είναι κάποια ισχυρή πιθανότητα (δηλαδή είναι μεγαλύτερη από την τιμή του threshold η μικρότερη από την τιμή της μονάδας μείον την τιμή του threshold) εισάγουμε στο input το concatenation τους.

Αυτή η διαδικασία επαναλαμβάνεται όσο η τιμή του threshold είναι μικρότερη από μία σταθερά και η τιμή του threshold αυξάνεται κατά την τιμή του stepvalue στο τελος της καθε επανάλιψης.

Εμείς έχουμε επιλέξει για το δυνατόν καλύτερο αποτέλεσμα την τιμή του threshold να ξεκινάει από 0.1 και να αυξάνεται κατά 0.1 μέχρι να φτάσει την τιμή 0.3.
Επίσης ο Sheduler αρχικοπειοίται ωστε να εκτελεί 16 threads και η τιμή του batch size είναι 1000.

Καταλήξαμε σε αυτές τις επιλογές μέσω πολλών πειραματισμών με διαφορετικές εκδοχές των τιμών οι οποίες φαίνονται και παρακάτω.

Στην συνεχεια μεσω της συνάρτησης Testing για κάθε ζευγάρι του αρχείου Testing.csv υπολογίζουμε την πρόβλεψη του εκπαιδευμενου μοντέλου και αφού αυτό γίνει για όλα τα ζευγάρια το πρόγραμμα εκτυπώνει το success rate του μοντέλου δηλαδή το ποσοστό των σωστων προβλέψεων

Τέλος αποδεσμευονται όλες οι δομές και το πρόγραμμα τερματίζει.

Δομές :

Η δομή Hash είναι η δομή που αποθηκευονται τα δεδομένα του κάθε json αρχείου.

Είναι ένας δυναμικός πίνακας κατακερματισμου με bucket-list όπου ως κλειδί χρησιμοποιεί το id κάθε json (πχ www.ebay.com//567)και κάνει rehash κάθε φορά που φτάνει 80% πληρότητα.

Για την υλοποίηση του bucket-list χρησιμοποιείται η δομή NList που σε κάθε κόμβο της αποθηκεύεται:

το id του json στην μεταβλητή camera τύπου char*

οι λέξεις του jsin αρχείου στην μεταβλητή spear που είναι τύπου Whash*

το αντίστοιχο vector του json που είναι τύπου HVector*

ένας δείκτης σε δομη CList που αντιστοιχεί στην κλικα την οποία ανήκει η camera.

Η δομή CList είναι μια συνδεδεμένη λίστα που για την υλοποίηση των κλικων.

Σε κάθε θέση της αποθηκεύει:

το όνομα της κάμερας

έναν δείκτη σε NList που αντιστοιχεί με τον κόμβο NList που είναι αποθηκευμένα τα στοιχεία της κάμερας.

Ο πρώτος κόμβος της κάθε CList δεν αποθηκεύει δεδομένα μιας κάμερας αλλά μια λίστα TList(συνδεδεμένη λίστα που αποθηκεύει δείκτες σε CList) που αποθηκεύει τις κλικες με τις οποίες δεν τεροιαζει η κλικα.

Η δομή WHash είναι μια δομή Πίνακα κατακερματισμου χωρίς bucket-list που αποφεύγει τα collision πηγαίνοντας στην επόμενη διαθέσιμη κενή θέση και κάνει rehash όταν έχει 80% πληρότητα.
Η δομή αυτή χρησιμοποιείται για να αποθηκεύει τις λέξης του κάθε json αρχείου με κλειδί την κάθε λέξη.
Σε κάθε bucket αποθηκεύει μια λέξη και το tf αυτής της λέξης για το συγκεκριμένο json.

Η δομή Hvector είναι μια δομή πίνακα κατακερματισμου χωρίς bucket-list που αποφεύγει τα collision πηγαίνοντας στην επόμενη διαθέσιμη κενή θέση και κάνει rehash όταν έχει 80% πληρότητα.
Η δομή αυτή αναπαριστά το vector της κάθε κάμερας.
Σε κάθε bucket αποθηκεύει την θέση και την tf*idf της αντίστοιχης θέσεις για κάθε λέξη απο τις 1000 πιο σημαντικές του λεφιλογίου ενα υπαρχει στην κάμερα.

Η δομή LHash είναι μια δομή Πίνακα κατακερματισμου χωρίς bucket-list που αποφεύγει τα collision πηγαίνοντας στην επόμενη διαθέσιμη κενή θέση και κάνει rehash όταν έχει 80% πληρότητα.
Η δομή αυτή χρησιμοποιείται για την υλοποίηση του λεξιλογιου όλων των json με κλειδί την κάθε λέξη.
Σε κάθε bucket αποθηκεύεται η κάθε λέξη το idf και το μέσω tf-idf της κάθε λέξης

Η δήλωση όλων των δομων λιστων της δομής Hvector και της δομής WHash και τα πρότυπα των συναρτήσεων για την διαχείριση τους βρίσκονται στο αρχείο list.h και οι υλοποιήσεις των συναρτήσεων στο αρχείο list.c

Η δήλωση όλων των δομων Hash και WHash και τα πρότυπα των συναρτήσεων για την διαχείριση τους βρίσκονται στο αρχείο hash.h και οι υλοποιήσεις των συναρτήσεων στο αρχείο hash.c

Η δήλωση της δομής Model και Input και τα πρότυπα των συναρτήσεων για την διαχείριση τους βρίσκονται στο αρχείο logistic.h και οι υλοποιήσεις των συναρτήσεων στο αρχείο logistic.c

Η δομή Job scheduler κατά την αρχικοποίηση της της δέχεται μία τιμή η οποία δείχνει το πόσα threads θα μπορεί να υποστηρίζει. Αρχικοποιει τις τιμές της και δημιουργεί τόσα threads όσα ζητήθηκαν τα οποία τρέχουν τη ρουτίνα worker.

Η ρουτίνα worker είναι στην ουσία η καρδιά του scheduler και λειτουργεί ως εξής :

Τρέχει μία while η οποία τερματίζει μόνο όταν καταστραφεί ο scheduler και μέσα σε αυτή την while κάθε φορά περιμένει να εμφανιστεί ένα καινούργιο job και μόλις εμφανιστεί το εκτελεί στη συνέχεια το καταστρέφει και μετα περιμένει να πάρει κάποιο καινύριο job.

Η δήλωση της δομής JobScheduler και job και τα πρότυπα των συναρτήσεων για την διαχείριση τους βρίσκονται στο αρχείο JobSheduler.h και οι υλοποιήσεις των συναρτήσεων στο αρχείο JobScheduler.c