Τσουράκης Ορφέας Α.Μ.: 1115201700175

Ταράτσας Ιωάννης Α.Μ.:1115201700160

Project K23α-3ο Μέρος Εργασίας

Ενδεικτικές Εκτελέσεις

Βασική εκτέλεση:

./project -x 2013_camera_specs -w sigmod_large_labelled_dataset.csv –s ./include/stopwords.txt

Unit Testing:

./all_tests

Το παραδοτέο στο τρίτο μέρος της εργασίας συμπεριλαμβάνει:

- Στον βασικό φάκελο, τα αρχεία main.c, structs.c, text_support.c, logistic_regression.c και το σχετικό makefile που παράγει τα παραπάνω εκτελέσιμα.
- Στον φάκελο tests, τα αρχεία stack_test.c, queue_test.c και rbt_test.c, heap_test.c, ht_test.c, text_test.c, thread_test.c, all_tests.c τα οποία χρησιμοποιούν την acutest.h, για το unit testing.
- Στον φάκελο include, έχουμε τα αρχεία structs.h, text_support.h, all_tests.h, logistic_regression.h, acutest.h, αλλά και ένα txt το οποίο συμπεριλαμβάνει τα stopwords τα οποία τα χρησιμοποιούμε για το καθάρισμα του κειμένου.

Λεπτομέρειες υλοποίησης:

main.c

Στην main.c προσθέσαμε στην αρχή μια ουρά nameList, που έχει όλα τα ονόματα των αρχείων και στο τέλος προσθέσαμε κώδικα για την επαναληπτική εκμάθηση. Ξεκινώντας, λοιπόν, την επαναληπτική εκμάθηση αρχικοποιούμε ένα HashTable (comb), το οποίο θα χρησιμεύσει για να κρατήσουμε τους συνδυασμούς του Training Set, αλλά και ένα Heap(newTrain), το οποίο θα περιλαμβάνει κάποιες νέες εγγραφές που θα ενσωματωθούν στο παλιό Training Set με βάση την πιθανότητα που θα προκύψει από το μοντέλο. Έτσι, αφού αρχικά καταστρέψουμε τις παλιές κλίκες, επαναφέρουμε τις κλίκες των Pairs(Item*, Clique*) και τις δημιουργούμε με βάση την πληροφορία που υπάρχει στο Training Set. Στην συνέχεια, αφού αρχικοποιήσουμε το μοντέλο, το προπονούμε με χρήση των threads και έπειτα, δημιουργείται το newTrain, το οποίο ενσωματώνουμε στο ήδη υπάρχων, κατασκευάζοντας έτσι στο τέλος το νέο Training Set. Αυτή η διαδικασία επαναλαμβάνεται

μέχρι το threshold να φτάσει το 0.5 ή μέχρι να τελειώσουν τα βήματα. Στην τελευταία επανάληψη εκτυπώνουμε μόνο τα related ζευγάρια, και πριν τερματίσει το πρόγραμμα το προπονούμε μια τελευταία φορά και εμφανίζουμε την απόδοση που προκύπτει με βάση το validation set. Το τεστ του validation set πραγματοποιείται και αυτό με την χρήση πολυνηματισμού.

structs.h

Προσθέσαμε την δομή Job Scheduler.

structs.c

Προστέθηκαν συναρτήσεις για τον Job Scheduler, συγκεκριμένα:

t_perror_exit(): Εκτύπωση μηνύματος λάθους για τα threads.

initialize_schelduler(): Αρχικοποίηση των mutexes και των condition variables του scheduler, αρχικοποίηση ενός κυκλικού buffer, που θα αποθηκεύει τα jobs, αρχικοποίηση του πίνακα με τα ids των threads και δημιουργία των threads.

add_job(): Καλείται μόνο από το main thread και αν υπάρχει χώρος στον buffer βάζει ένα job και στέλνει σήμα ότι εισάχθηκε ένα job, αλλιώς περιμένει να αδειάσει χώρος.

get_job(): Εκτελεί την αντίστροφη διαδικασία από την add_job() και καλείται από τα working threads για να πάρουν ένα job από το buffer.

destroy_schelduler(): Καταστροφή του κυκλικού buffer, των mutexes και των condition variables και του πίνακα με τα ids των threads.

Επιπλέον συναρτήσεις

RestorePairs(): Με αυτή την συνάρτηση αναδρομικά διασχίζουμε το Hash Table των Pairs, όπου πηγαίνουμε σε κάθε Pair και το αρχικοποιούμε ξανά.

text_support.c

TrainingSetStats(): Με αυτήν την συνάρτηση, διαβάζονται τα ζευγάρια από την ουρά του Train Set δημιουργούνται οι τελικές κλίκες.

CreateAllPairs(): Σε αυτήν την συνάρτηση παίρνουμε όλα τα ζευγάρια τα οποία μπορούν να δημιουργηθούν και αφού ελέγχουμε ότι δεν υπάρχουν στο τρέχον training set, μέσω της χρήσης του HashTable comb, τότε κάνουμε προβλέψεις με βάση το μοντέλο και αν η πιθανότητα είναι μικρότερη του threshold βάζουμε στο record την τιμή 0 και εισάγουμε το

record αυτό με την αντίθετη *τιμή της πιθανότητας* στο Max Heap(αρνητική τιμή λόγω max heap και επειδή θέλουμε να εξάγουμε την μικρότερη τιμή). Αντίστοιχα, αν η πιθανότητα είναι μεγαλύτερη του 1-threshold βάζουμε στο record την τιμή 1 και εισάγουμε το record αυτό με την αντίθετη τιμή της 1- *τιμή της πιθανότητας* στο Max Heap(αρνητική τιμή λόγω max heap και επειδή θέλουμε να εξάγουμε την μικρότερη τιμή).

CreateNewTrainingSet(): Με αυτήν την συνάρτηση συγχρονίζουμε και αναθέτουμε jobs στα threads που εκτελούν την παραπάνω συνάρτηση.

unrelatedFound(): Η συνάρτηση αυτή επιστρέφει 1 αν κάποιο στοιχείο στην related λίστα βρεθεί στο Hash Table unrelated και 0 αν δεν βρεθεί.

ResolveTransitivity(): Με αυτήν την συνάρτηση ενσωματώνουμε τα στοιχεία που εξάγονται από το Heap newTrain, στα Pairs στα οποία έχουν ήδη δημιουργηθεί, πιο συγκεκριμένα ελέγχουμε πάντα αν δύο στοιχεία βρίσκονται σε διαφορετικές κλίκες. Αν ταιριάζουν και ανάμεσα στις κλίκες τους δεν υπάρχει αρνητική συσχέτιση τότε ενώνουμε τις κλίκες, αλλιώς κάνουμε την κάθε κλίκα να δείχνει στο στοιχείο(Pair) που δεν το περιέχει (δηλαδή που δεν ταιριάζει).

logistic_regression.c

Στο logistic_regression.c προστέθηκαν αλλά και τροποποιήθηκαν συναρτήσεις, έτσι ώστε να χρησιμοποιούν threads, για την επίτευξη ενός πιο γρήγορου train στο μοντέλου.

LRthreadTrain(): Η συνάρτηση αυτή προπονεί το μοντέλο με την χρήση threads, σύμφωνα με το post που ανέβηκε στο piazza.

LRtrain(): Η συνάρτηση αυτή πλέον δέχεται και έναν αριθμό από threads, που θα χρησιμοποιήσουμε, αλλά και το μέγεθος του batch. Έτσι, αφού αρχικοποιήσουμε, έναν scheduler, στέλνουμε κάθε φορά έναν συγκεκριμένο αριθμό από εγγραφές συγχρονίζοντας παράλληλα και τα threads. Με αυτόν τον τρόπο, καταφέρνουμε να χωρίσουμε μια εργασία σε επιμέρους εργασίες, κάνοντας έτσι το πρόγραμμα πιο γρήγορο αλλά και αποδοτικό.

LRthreadTest(): Η συνάρτηση αυτή τεστάρει το μοντέλο με την χρήση threads.

LRtest(): Η συνάρτηση αυτή πλέον δέχεται και έναν αριθμό από threads, που θα χρησιμοποιήσουμε, αλλά και το μέγεθος του batch. Η λογική της είναι παρόμοια με αυτή της LRtrain(), αλλά πολύ πιο απλή.

Φάκελος Tests

Έχει προστεθεί το αρχείο thread_test.c το οποίο περιλαμβάνει τα αντίστοιχα test για τα threads.