Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα

Χειμερινό Εξάμηνο 2023-2024

Τελική Αναφορά

Εζεκελιάν Αναΐτ (sdi1900056)
Παπασωτήρη Χριστινα (sdi1900151)

1. Εισαγωγή

Η αναζήτηση εγγύτερου γείτονα (Nearest Neighbor Search - NSS) αντιπροσωπεύει ένα πρόβλημα βελτιστοποίησης που αποσκοπεί στον εντοπισμό του σημείου ενός συνόλου δεδομένων που είναι πιο κοντινό ή πιο όμοιο με ένα δεδομένο σημείο. Η έννοια της εγγύτητας εκφράζεται μέσω μιας μετρικής ανομοιότητας, όπου χαμηλή τιμή υποδηλώνει υψηλή ομοιότητα.

Συνήθως, το πρόβλημα αναζήτησης εγγύτερου γείτονα ορίζεται ως εξής: δεδομένο ένα σύνολο σημείων S σε έναν χώρο M και ένα σημείο επερώτησης $q \in M$, η ερώτηση είναι να βρείτε το σημείο του S που είναι το πλησιέστερο στο q. Η ανομοιότητα συνήθως μετριέται μέσω μιας μετρικής απόστασης σε έναν μετρικό χώρο M.

Μια γενίκευση αυτού του προβλήματος είναι η αναζήτηση των k εγγύτερων γειτόνων (K-Nearest Neighbors - KNN), όπου αναζητούνται τα k πλησιέστερα σημεία σε ένα δεδομένο σημείο. Ένα συνήθης τρόπος υλοποίησης είναι μέσω του γράφου εγγύτερων γειτόνων (k-Nearest Neighbors Graph - KNNG), όπου κάθε κορυφή του γράφου συνδέεται με τα k πλησιέστερα σημεία.

Η κατασκευή του KNNG μέσω τεχνικών brute-force είναι χρονοβόρα και πρακτικά εφαρμόσιμη μόνο για μικρά σύνολα δεδομένων.

Για το λόγο αυτό, σε αυτή την εργασία, προσπαθήσαμε να υλοποιήσουμε έναν αλγόριθμο για την προσεγγιστική λύση του προβλήματος στοχεύοντας σε καλύτερη ταχύτητα με την καλύτερη δυνατή αποτελεσματικότητα συγκριτικά με τον brute-force αλγόριθμο.

Ο αλγόριθμος που χρησιμοποιήσαμε είναι ο NN-Descent.
Ο βασικός αλγόριθμος κατασκευής του γράφου έχει ως εξής:
1. Υλοποίησε έναν τυχαίο γράφο (σύνδεσε κάθε κόμβο με K τυχαίους κόμβους
2. Για κάθε κόμβο: Μέτρησε την απόσταση από τον κόμβο στους γείτονες των γειτόνων Αν κάποιοι γείτονες γειτόνων είναι εγγύτεροι, ενημέρωσε τον γράφο ανάλογα και διατήρησε μόνο τους K εγγύτερους γείτονες
3. Αν τροποποιήθηκε ο γράφος πήγαινε στο βήμα 2, αλλιώς τερμάτισε.

Στην εργασία αυτή υλοποιήσαμε 3 διαφορετικές εκδοχές, που αποτελούν η μία βελτιστοποίηση της άλλης και πιο συγκεκριμένα:

- Εκδοχή 1η
  Υλοποίηση του βασικού αλγορίθμου για dataset με σημεία 100 διαστάσεων που μας δόθηκαν και 2 διαστάσεων που κατασκευάσαμε εμείς. Υλοποίηση του αλγορίθμου Brute Force και αποθήκευση των αποτελεσμάτων σε binary αρχείο για σύγκριση με τον NN-Descent. Υπολογισμός των αποστάσεων με τη

χρήση δύο διαφορετικών συναρτήσεων υπολογισμού απόστασης (Ευκλείδεια και Μανχάταν) και unit test σε όλες τις public συναρτήσεις των δομών.

- Εκδοχή 2η
    - Βελτιστοποίηση του βασικού αλγορίθμου με τις εξής τεχνικές:
    - Local Join
    - Σταδιακή Αναζήτηση(Incremental Search)
    - Δειγματοληψία
    - Πρόωρος Τερματισμός
- Εκδοχή 3η
    - Βελτιστοποίηση του υπολογισμού της απόστασης με τη χρήση της Ευκλείδειας Νόρμας.
    - Αρχικοποίηση του KNN γράφου με τη μέθοδο των δένδρων τυχαίων προβολών (random projection trees)
    - Παράλληλη εκτέλεση με χρήση νημάτων

2. Μοντελοποίηση/Υλοποίηση

- Επιλογές δομών δεδομένων που χρησιμοποιήθηκαν.
    - Χρησιμοποιήσαμε templates για να έχουμε πιο ευέλικτο και αποτελεσματικό κώδικα τον οποίο μπορούμε να επαναχρησιμοποιήσουμε για διάφορες δομές δεδομένων. Η προσαρμογή σε διαφορετικούς τύπους δεδομένων ήταν απαραίτητο εργαλείο.
    - Το vector είναι μια δυναμική υλοποίηση πινάκων.
    - Ο graph είναι μια συλλογή κόμβων που συνδέονται με ακμές.
    - Το vertex είναι ο κόμβος της δομής γράφου.
    - Η dll είναι μια διπλά συνδεδεμένη λίστα που αποτελείται από κόμβους. Κάθε κόμβος περιέχει δεδομένα και δύο δείκτες, έναν προς τον προηγούμενο κόμβο και έναν προς τον επόμενο.
    - Η queue είναι μια δομή που λειτουργεί σύμφωνα με την αρχή: first in - first out.
    - Ο job_scheduler είναι μια δομή που διαχειρίζεται την εκτέλεση διαφόρων εργασιών με τη χρήση μια δεξαμενής νημάτων, σημαφόρων και μιας ουράς προτεραιότητας η οποία περιέχει τα job που θέλουμε να εκτελεστούν παράλληλα.
    - Η job είναι μια ρουτίνα κώδικα η οποία θέλουμε να εκτελεστεί από κάποιο νήμα πιθανότατα παράλληλα από κάποια άλλη.
    - Η point_norm_job είναι η ρουτίνα κώδικα που υπολογίζει την νόρμα δύο σημείων.
    - Το RPTree είναι η δομή για την αναπαράσταση των δένδρων τυχαίων προβολών, που χρησιμοποιούμε για την αρχικοποίηση του γράφου. Αποτελείται από RPTreeNode, δηλαδή κόμβους στους οποίους αποθηκεύονται πίνακες από πίνακες (δηλαδή σημεία πολλών

διαστάσεων) και δείκτες στον αριστερό και δεξί τους κόμβο-παιδί, οι οποίοι δημιουργούνται κατά τη διαδικασία διαχωρισμού του dataset σε μικρότερα. Η δομή RPTree περιέχει έναν δείκτη σε RPTreeNode που είναι η ρίζα του δένδρου.

- Προβλήματα και προκλήσεις που αντιμετωπίσατε.
    - Δυσκολίες στην κατανόηση του τρόπου χρήσης της βιβλιοθήκης Catch2 για τον έλεγχο ορθότητας λογισμικού.
    - Δυσκολίες στην σύνδεση του Github actions με το cmake (χρήση για την μεταγλώττιση των αρχείων) για έλεγχο με αυτοματοποιημένο τρόπο σε κάθε push με τα test που υλοποιήσαμε.
    - Δυσκολία στην κατασκευή ενός τρόπου διαχείρισης διαφορετικών συναρτήσεων που θα εκτελούνται παράλληλα.
    - Δυσκολία στον συγχρονισμό των νημάτων και συγκεκριμένα του κλειδώματος και ξεκλειδώματος των νημάτων για την εκτέλεση της εργασίας.

3. Βελτιστοποιήσεις

- Local Join
  Η διαδικασία κατασκευής του γράφου αρχικά περιλαμβάνει τη δημιουργία ενός συνόλου "γειτόνων" για κάθε κόμβο, τόσο των άμεσων όσο και των αντίστροφων γειτόνων. Στη συνέχεια, η αναβάθμιση του γράφου πραγματοποιείται μέσω του υπολογισμού όλων των αποστάσεων μεταξύ των ζευγαριών γειτόνων για κάθε σύνολο άμεσων γειτόνων.
  Μετά το πρώτο βήμα, όπου υπολογίζονται οι αντίστροφοι γείτονες, ο υπολογισμός μπορεί να εκτελεστεί τοπικά για κάθε κόμβο. Κατά τη διάρκεια αυτής της διαδικασίας, αποθηκεύονται προσωρινά οι απαραίτητες τροποποιήσεις για κάθε σύνολο γειτόνων, και οι υπολογισμοί εκτελούνται στο τέλος του βήματος επανάληψης.
  Αυτή η τεχνική δεν επηρεάζει τον ίδιο τον αλγόριθμο, αλλά επιτρέπει την παρακολούθηση των κόμβων που έχουν "νέους" γείτονες και χρειάζονται υπολογισμούς απόστασης από τους υπάρχοντες γείτονες.

- Σταδιακή Αναζήτηση(Incremental Search)
  Σε κάθε επανάληψη, με στόχο την αποφυγεί των πολλαπλών ίδιων συγκρίσεων σε κάθε τοπικό join, προσθέτουμε στους κόμβους μια boolean σημαία, η οποία αλλάζει σε false όταν ο κόμβος αυτός συμμετέχει σε τοπικό join. Επομένως, χρησιμοποιούμε για τις συγκρίσεις, μόνο τους κόμβους που η σημαία τους έχει τιμή true και δεν έχουν ακόμα συγκριθεί.

- Δειγματοληψία

Πριν από κάθε τοπικο join, επιλέγονται pK, p ∈ [0,1] τυχαίοι κόμβοι από του διαθέσιμους, δηλαδή όσους έχουν την boolean σημαία τους ίση με true, για να συμμετέχουν στις συγκρίσεις. Οι αντίστροφοι γείτονες αντιμετωπίζονται κι αυτοί σε ξεχωριστή λίστα συνεπώς είναι κι αυτοί το πολύ pK.

- Πρόωρος Τερματισμός
  Αντί να τερματίζει ο αλγόριθμος όταν δεν υπάρχουν καθόλου αλλαγές στο γράφο, μια τεχνική για την ταχύτερη εκτέλεσή του, είναι να τερματίζει όταν οι αλλαγές που γίνονται είναι πολύ λίγες.
- Βελτιστοποίηση του υπολογισμού της απόστασης με τη χρήση της Ευκλείδειας Νόρμας η οποία εκτελείται παράλληλα
- Αρχικοποίηση του KNN γράφου με τη μέθοδο των δένδρων τυχαίων προβολών (random projection trees)
  Η μέθοδος αυτή, χωρίζει το αρχικό dataset σε μικρότερα, με μέγεθος το πολύ D. Ουσιαστικά για κάθε dataset, δημιουργούμε ένα υπερεπίπεδο ανάλογο των διαστάσεων των σημείων (πίνακα με τυχαίες τιμές) και υπολογίζοντας το εσωτερικό γινόμενο του υπερεπιπέδου αυτού με κάθε σημείο, το χωρίζουμε σε δύο υποσύνολα, ένα μικρότερο κι ένα μεγαλύτερο από μια τυχαία τιμή c.
- Παράλληλη εκτέλεση με χρήση νημάτων και ουράς προτεραιότητας. Στο κύριο πρόγραμμα καταχωρούνται εργασίες (jobs) στον job scheduler προσθέτοντας μια εργασία που δημιουργήθηκε στην ουρά εργασιών.
  Η εκκίνηση της εκτέλεσης γίνεται με τη συνάρτηση start_execute του job scheduler, χρησιμοποιούνται πολλαπλά νήματα κάθε ένα από τα οποία εκτελεί τη συνάρτηση work_thread.
  Κάθε νήμα όσο είναι ενεργό λαμβάνει μια εργασία από την ουρά εργασιών jobs.
  Εάν η ουρά εργασιών είναι διαθέσιμη, το νήμα αφαιρεί την εργασία από την ουρά και απελευθερώνει το mutex.
  Η εργασία που πήρε το νήμα εκτελείται με την εκτέλεση της execute.
  Μετά την ολοκλήρωση της εκτέλεσης, η εργασία διαγράφεται.
  Τα νήματα συνεχίζουν να εκτελούνται μέχρι να τερματιστούν.
  Η διαδικασία αυτή επαναλαμβάνεται για κάθε εργασία που καταχωρείται.

4. Αναφορές

https://www.researchgate.net/publication/330397378_K-nearest_Neighbor_Search_by_Random_Projection_Forests