Διαχείριση Χωρο-κειμενικών Δεδομένων Μεγάλης Κλίμακας

Αντώνης Ψαρρός

Επιβλέπων: καθηγητής Χρήστος Δουλκερίδης

Μεταπτυχιακό Πρόγραμμα "Πληροφοριακά Συστήματα & Υπηρεσίες"
Ειδίκευση "Μεγάλα Δεδομένα και Αναλυτική"
Τμήμα Ψηφιακών Συστημάτων
Πανεπιστήμιο Πειραιά

Φεβρουάριος 2021

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, κ. Δουλκερίδη Χρήστο, που με τις πολύτιμες υποδείξεις και παρατηρήσεις του, συνέβαλε στη διαμόρφωση μιας πιο ολοκληρωμένης εικόνας ως προς την προσέγγιση του θέματος: “Διαχείριση Χωροκειμενικών Δεδομένων Μεγάλης Κλίμακας”, και διεύρυνε τους ορίζοντές μου και το ενδιαφέρον μου για την έρευνα.

Τέλος, θα ήθελα να ευχαριστήσω θερμά τους γονείς μου για την ψυχολογική και υλική υποστήριξη που μου παρείχαν.

Περίληψη

Κατά την εκπόνηση της παρούσας Διπλωματικής εργασίας πραγματεύθηκε το πρόβλημα της σύζευξης μεγάλου όγκου χωροκειμενικών δεδομένων. Δεδομένα τα οποία αποτελούνται από εγγραφές όπου η κάθε μία περιέχει (lon,lat,text) όπου lon, lat είναι οι συντεταγμένες και text είναι το κείμενο κάθε εγγραφής, κάθε εγγραφή αποτελεί ένα αντικείμενο x. Το κριτήριο για την ομοιότητα δύο αντικείμενων x,v είναι ο έλεγχος της απόστασης στο χώρο των x,v το οποίο υπολογίζεται με τον τύπο Haversine Distance. Για την εύρεση της ομοιότητας του κειμένου των δύο αντικειμένων x,v χρησιμοποιείται το τύπος Jaccard Similarity.

Για το παραπάνω πρόβλημα δημιουργήθηκε ένας αλγόριθμος που το υλοποιεί και έγινε εφαρμογή αυτού του αλγορίθμου σε κατανεμημένο περιβάλλον από υπολογιστές με χρήση του εργαλείου Apache Spark, όπου χρησιμοποίει τις δυνατότητες επεξεργασίας δεδομένων του επεξεργαστή κάθε υπολογιστή. Επίσης, έγινε μια επιπλέον υλοποίηση του αλγορίθμου με σκοπό να εκμεταλλεύεται τις δυνατότητες επεξεργασίες της κάρτας γραφικών κάτι που επιτεύχθηκε χρησιμοποιώντας εργαλεία της Nvidia.

Τέλος, με για την εκτέλεση αλγορίθμων χωροκειμενικής σύζευξης σε Apache Spark δημιουργήθηκε διαδικτυακή πλατφόρμα με σκοπό να έχει τη δυνατότητα ένας χρήστης να ανεβάζει έναν αλγόριθμο που απαντά σε αυτό το πρόβλημα, να τον εκτελεί και να κάνει προβολή των αποτελεσμάτων σε χάρτη. Επιπλέον, ο χρήστης μπορεί να κάνει προβολή των dataset που δέχεται σαν είσοδο ο κάθε αλγόριθμος σε χάρτη.

Abstract

In this thesis attempt was made to deal with the problem of joining a great amount of spatio-textual data. Data consist of records each one of which consists of the coordinates (lon, lan) and the text. Every record is an object x. There are two criteria that define the similarity between two objects, the distance and the text similarity.

In order for the problem mentioned above to be solved, an algorithm was created. This algorithm was put into practice in distributed computer environment, using the Apache Spark tool and taking advantage of the power and the data processing abilities of every computer processor. The algorithm was also used with the aim of taking advantage of the graphics processor abilities which was managed with the use of Nvidia tools.

Finally, an online platform was created for the execution of spatio-textual join algorithms in Apache Spark. In this platform each user is able to upload an algorithm that solves the problem, execute it, and even visualize the results in a chart. Moreover, it offers the ability to visualize the initial dataset that the algorithm accepts as an input.

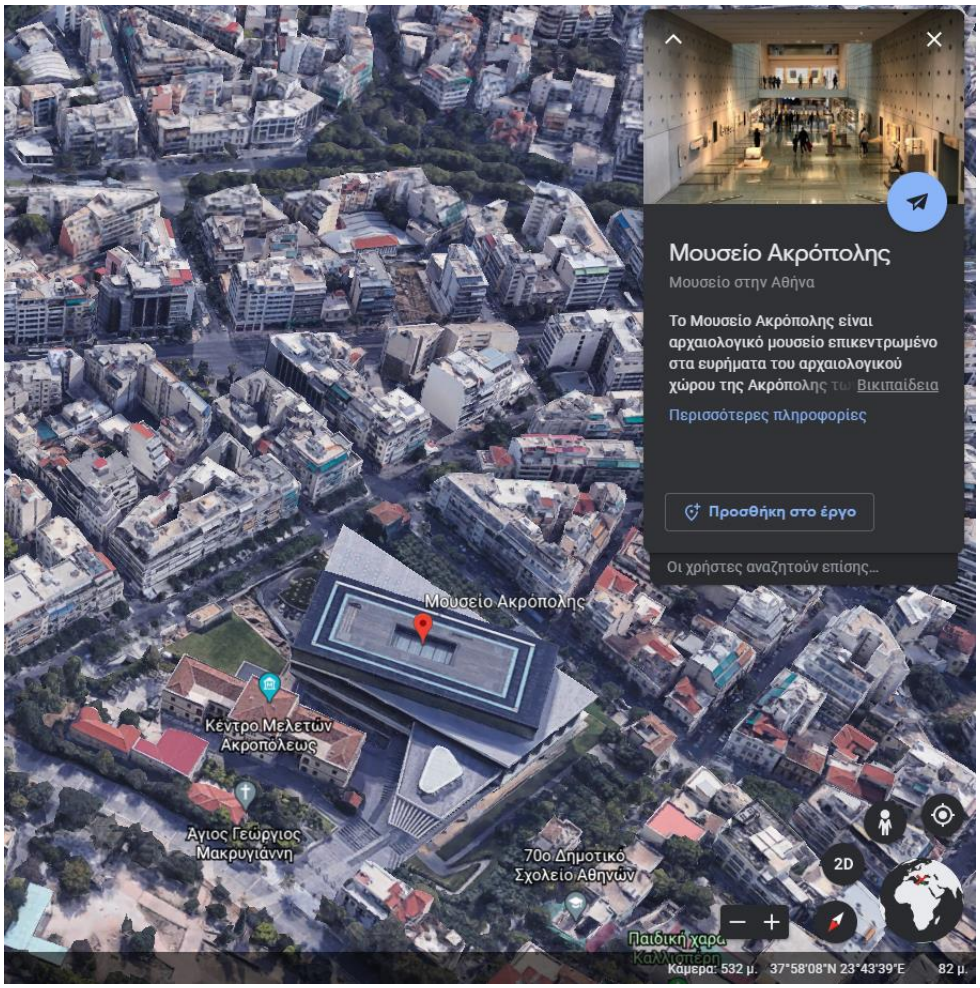
Περιεχόμενα

Περιεχόμενα Εικόνων

Περιεχόμενα Πινάκων

1 Κεφάλαιο – Εισαγωγή

Στις μέρες μας, λόγω τις αυξημένης χρήσης του GPS των φορητών συσκευών όπως τα κινητά τηλέφωνα και τα tablets, υπάρχει παραγωγή αυξημένου όγκου χωρο-κειμενικών (spatio-textual) δεδομένων, τα οποία περιέχουν γεωγραφικές συντεταγμένες και πληροφορία κειμένου.



Εικόνα 1 Χωρο-κειμενικά Δεδομένα Πηγή: Google Earth

Ένα απτό παράδειγμα χωρο-κειμενικών δεδομένων είναι η παραπάνω εικόνα από το Google Earth. Παρατηρούμε ότι το επιλεγμένο sitemap που απεικονίζει το μουσείο της Ακρόπολης, περιέχει χωρική πληροφορία που προβάλλεται κάτω δεξιά και αφορά τις συντεταγμένες του μουσείου. Επίσης, η περιγραφή σχετικά με το τι βρίσκεται στο χώρο του Μουσείου που παρουσιάζεται στο αναδυόμενο παράθυρο που προκύπτει μετά από click στο Μουσείο της Ακρόπολης, αποτελεί πληροφορία κειμένου.

Το παραπάνω είναι ένα παράδειγμα εφαρμογής που περιέχει χωρο-κειμενικά δεδομένα. Μερικές περαιτέρω εφαρμογές που επίσης χρησιμοποιούν τέτοιου είδους δεδομένα είναι το Facebook όταν κάποιος ανεβάζει ένα post όπου περιέχεται κείμενο και παράλληλα προβάλλονται και οι συντεταγμένες

που είχε ο χρήστης όταν ανέβασε την σχετική ανακοίνωση. Το Twitter και το Instagram αποτελούν παρόμοιου τύπου εφαρμογές.

Τα δεδομένα αυτά που παράγουν οι παραπάνω εφαρμογές είναι μεγάλου όγκου και εντάσσονται στην ενότητα Big Data. Για την διαχείρισή τους απαιτείται μεθοδευμένη διαχείριση σε συστήματα που μπορούν να υποστηρίξουν μιας τόσο μεγάλης κλίμακας δεδομένα. Βέβαια η διαχείριση κρίνεται απαραίτητη διότι μέσα από αυτή μπορούν να απαντηθούν ερωτήματα τα οποία αν έπρεπε να τα επεξεργαστεί κάποιος σε έναν υπολογιστή ενδέχεται να μην μπορούσαν να εκτελεστούν ή χρόνος εκτέλεσης να ήταν απαγορευτικά μεγάλος. Για παράδειγμα, ένα ερώτημα το οποίο θα ήταν άξιο υλοποίησης σύμφωνα με την προηγούμενη εικόνα είναι: «Πόσα μουσεία στην Ευρώπη έχουν παρόμοια περιγραφή με την Ακρόπολη;». Επειδή ο όγκος των δεδομένων που θα επεξεργαστούν για να απαντηθεί το παραπάνω ερώτημα είναι μεγάλος, θα χρειαστεί η επεξεργασία του να υλοποιηθεί σε μεγαλύτερης κλίμακας συστήματα από έναν υπολογιστή και ενδεχομένως να χρειαστεί ακόμα και ένα δίκτυο με περισσότερους από έναν υπολογιστές που θα κάνουν παράλληλα την επεξεργασία των δεδομένων.

1.1 Χωρο-κειμενική Σύζευξη (Spatio-Textual Join)

Για να παραχθεί κάποιο αποτέλεσμα από τα χωρο-κειμενικά δεδομένα μια από τις επικρατέστερες τεχνικές που εφαρμόζονται πάνω σε αυτά είναι η υλοποίηση κάποιων ερωτημάτων. Για παράδειγμα ένα ερώτημα θα μπορούσε να αποτελέσει το ακόλουθο: «Επίστρεψε αυτά τα αντικείμενα που έχουν απόσταση μικρότερη από x και κειμενική ομοιότητα μεγαλύτερη από ψ». Για την υλοποίηση αυτού του ερωτήματος θα έπρεπε να εκτελεστεί η παρακάτω υλοποίηση:

- Ανάγνωση των δεδομένων
- Δημιουργία ή ανάγνωση των ετικετών για τα διαφορετικά είδη δεδομένων που περιλαμβάνονται στο dataset. Για παράδειγμα δύο είδη ετικετών θα μπορούσαν να είναι το a, b.
- Αντιστοίχιση όλων των αντικειμένου του είδους a με αυτά του είδους b για την εύρεση των αποστάσεών τους και της κειμενικής ομοιότητας τους. Η συγκεκριμένη διαδικασία ονομάζεται cross join query διότι συσχετίζουμε (join) όλα τα αντικείμενα του είδους a με αυτά του είδους b (cross).
- Στα δεδομένα που παράχθηκαν από την διαδικασία του cross join υλοποιείται ένα ακόμα query για την επιστροφή των ζευγαριών από αντικείμενα τα οποία τώρα πλέον έχουν μια προκαθορισμένη απόσταση και μια προκαθορισμένη κειμενική ομοιότητα.

1.2 Περίληψη βιβλιογραφίας για την Χωρο-κειμενική Σύζευξη

Για την δημιουργία αλγορίθμου που να απαντά στο ερώτημα της χωρο-κειμενικής σζευξης προηγήθηκε έρευνα στην αντίστοιχη βιβλιογραφία . Το ερώτημα αυτό λοιπόν χωρίζεται σε δύο βασικούς τύπους στην σχετική αυτή έρευνα:

- **Spatio-Textual Similarity:** όπου υπολογίζεται και επιστρέφεται η χωρική και κειμενική ομοιότητα από κάθε ζεύγος αντικειμένων που συνάδει με τις δύο προκαθορισμένες τιμές που έχει προκαθορίσει ο χρήστης για την κειμενική ομοιότητα και για την χωρική απόσταση των αντικειμένων.
- **Top-k Spatio-Textual Similarity Join**: όπου υπολογίζονται και επιστρέφονται τα k ζευγάρια αντικειμένων με την μεγαλύτερη χωρική και κειμενική ομοιότητα. Ο χρήστης ορίζει τον αριθμό k των ζευγαριών που επιθυμεί να επιστραφούν.

Ένα θεμελιώδες κριτήριο για την αποδοτικότητα του αλγορίθμου είναι ο τρόπος διαμερισμού των δεδομένων με στόχο να επιτευχθεί η βέλτιστη παράλληλη επεξεργασία αυτών. Η διαδικασία αυτή ονομάζεται partitioning που επίσης χωρίζεται στις ακόλουθες δύο βασικές κατηγορίες.

- **Spatial First:** Οι αλγόριθμοι που εφαρμόζουν την τεχνική Spatial First σημαίνει πως επεξεργάζονται πρώτα τα δεδομένα βάσει των χωρικών τους κριτηρίων και στη συνέχεια με βάση την κειμενική τους ομοιότητα. Δηλαδή ο διαμερισμός των δεδομένων (partitioning) θα γίνει βάσει του χώρου σε μικρότερα χωρία.
- **Textual First:** Οι αλγόριθμοι που εφαρμόζουν την τεχνική Textual First επεξεργάζονται πρώτα την πληροφορία κειμένου των δεδομένων και μετά προχωρούν στα χωρικά κριτήρια. Ένας αλγόριθμος που αξιοποιεί την κειμενική πληροφορία για να δημιουργήσει μια τεχνική partitioning θα ήταν ο εξής: Έστω ότι έχουμε 2 εγγραφές με πέντε λέξεις η κάθε εγγραφή αν οι δύο εγγραφές έχουν μια κοινή λέξη από τις δύο πρώτες κάθε εγγραφής τότε οι εγγραφές αυτές θα μπουν στο ίδιο partition για περεταίρω επεξεργασία. Αλλιώς θα μπουν σε διαφορετικό διότι δεν έχουν πιθανότητα να καλύπτουν τα κριτήρια της κειμενικής ομοιότητας.

Τέλος έχουν μελετηθεί και συγκρίσεις μεταξύ των αλγορίθμων που εκτελούνται κεντρικοποιημένα σε ένα υπολογιστή σε σχέση με αυτούς που εκτελούνται σε ένα δίκτυο από υπολογιστές.

1.3 Στόχος

Στην συγκεκριμένη Διπλωματική οι στόχοι είναι τρεις:

- Σχεδιασμός αποδοτικού αλγορίθμου που να ικανοποιεί το ερώτημα του Spatio-Textual Join. Στην συγκεκριμένη περίπτωση εφαρμόστηκε η τεχνική του Spatial First.
- Σύγκριση επιδόσεων αλγορίθμου που να ικανοποίει το ερώτημα που Spatial Join τόσο για την εκτέλεση του στην κάρτα γραφικών όσο και σε περιβάλλον Apache Spark, το οποίο να αξιοποιεί τις δυνατότητες ενός δικτύου από υπολογιστές.
- Τέλος, δημιουργία διαδικτυακής πλατφόρμας για την εκτέλεση αλγορίθμων και οπτικοποίηση των αποτελεσμάτων τους. Οι οποίοι χρησιμοποιούν το περιβάλλον Apache Spark και εκτελούν Spatio-Textual Join.

1.4 Διάρθρωση κεφαλαίων της διπλωματικής

Κεφάλαιο 2: Βιβλιογραφική έρευνα σε αλγορίθμους που εκτελούν Spatio-Textual Similarity και Top-k Spatio Textual Similarity τόσο σε κεντρικοποιημένο περιβάλλον όσο και σε κατανεμημένο περιβάλλον. Εφαρμόζοντας διαφορετικούς τύπους partitioning και διαφορετική σειρά επεξεργασίας των δεδομένων είτε Spatial First είτε Textual First.

Κεφάλαιο 3: Επεξήγηση της δημιουργίας του δικού μας αλγορίθμου που ικανοποιεί το Spatio-Textual Join, προβολή πειραματικής διαδικασίας σε περιβάλλον Apache Spark και τέλος συγγραφή των συμπερασμάτων που προκύπτουν μέσα από την πειραματική διαδικασία.

Κεφάλαιο 4: Επεξήγηση της δημιουργίας αλγορίθμου που εκτελεί Spatial Join και εκτέλεση αυτού τόσο στην κάρτα γραφικών όσο και σε περιβάλλον Apache Spark. Προβολή της πειραματικής διαδικασίας και συγγραφή συμπερασμάτων που προκύπτουν από τα διαφορετικά περιβάλλοντα εκτέλεσης.

Κεφάλαιο 5: Δημιουργία διαδικτυακής πλατφόρμας με σκοπό να μπορεί ο χρήστης να ανεβάσει έναν αλγόριθμο και τα απαραίτητα dataset ώστε να τον εκτελέσει σε περιβάλλον Apache Spark και να κάνει οπτικοποίηση τόσο των dataset όσο και των αποτελεσμάτων σε χάρτες.

Κεφάλαιο 6: Συμπεράσματα από την εκπόνηση όλης της εργασίας και προτροπή επίλυσης δισεπίλυτων ερωτημάτων που προέκυψαν κατά την εκπόνηση της εργασίας.

2 Κεφάλαιο - Βιβλιογραφική έρευνα αλγορίθμων Χωρο-κειμενικής Σύζευξης

Στην παρακάτω έρευνα όλοι οι αλγόριθμοι που θα μελετήσουμε απαντούν στο ερώτημα της χωρο-κειμενικής σύζευξης. Βέβαια, ποικίλουν ως προς τον τρόπο που διαχειρίζονται τα δεδομένα, τον τύπο συστημάτων που εκτελούνται(κεντρικοποιημένα ή σε δίκτυο από υπολογιστές) και τον τρόπο που επιστρέφουν τα αποτελέσματα. Όλοι οι αλγόριθμοι έχουν κοινό κριτήριο σύζευξης. Για τις αποστάσεις των αντικειμένων χρησιμοποιούν την ευκλείδεια απόσταση και για την ομοιότητα των κειμένων χρησιμοποιούν την Jaccard και για τις δύο από τις οποίες ορίζεται ένα κατώφλι απόστασης από τον χρήστη αλλά και ένα κατώφλι ομοιότητας κειμένου. Αναλυτικά παρατίθενται παρακάτω.

Η εκτέλεση αυτής της σύζευξης μπορεί να εκτελεστεί είτε σε κεντρικοποιημένο περιβάλλον υπολογιστών είτε σε κατανεμημένο. Σκοπός μας είναι η εκτέλεση σε κατανεμημένο περιβάλλον υπολογιστών λόγω των αυξημένων απαιτήσεων που έχουν τα δεδομένα λίγο μεγαλύτερης κλίμακας των οποίων η εκτέλεσή τους σε έναν υπολογιστή επιφέρει καθυστερήσεις. Κατά την διάρκεια της μελέτης για τους αλγορίθμους επίλυσης του συγκεκριμένου προβλήματος, οι αλγόριθμοι που μελετήθηκαν χωρίζονται σε 3 βασικά κομμάτια:

- **Indexing** (δημιουργία ευρετηρίου για τα δεδομένα μας)
- **Partitioning** (τρόποι διαχωρισμού των δεδομένων για να διαμοιραστούν στους υπολογιστές)
- **Grouping Objects** (τρόποι ομαδοποίησης των δεδομένων για την επίτευξη πιο αποδοτικού υπολογισμού κατά την διάρκεια του join)
- **Similarity join query** (εκτέλεση της σύζευξης των δεδομένων ανάλογα με τα όρια που έχουν οριστεί)

2.1 Indexing

2.1.1 Textual Objects

Για να βρεθεί η ομοιότητα των αντικειμένων x μιας συλλογής R θα πρέπει να ελεγχθεί η ομοιότητα κάθε αντικειμένου x με ένα άλλο αντικείμενο ψ. Αρχικά για όλο το R έχουμε μια λίστα (posting list) που περιέχει όλες τις λέξεις που εμφανίζονται σε όλα τα αντικείμενα και πόσες φορές εμφανίζονται σε κάθε αντικείμενο. Οπότε έτσι έχουμε μια λίστα για το R που περιλαμβάνει την ποικιλία των λέξεων που εμφανίζονται σε μια συλλογή από αντικείμενα και πόσες φορές εμφανίζονται σε κάθε αντικείμενο.

Λέξεις Συλλογής Αντικειμένων	Συχνότητα εμφάνισης λέξεων στο αντικείμενο Α	Συχνότητα εμφάνισης λέξεων στο αντικείμενο Β	Συχνότητα εμφάνισης λέξεων στο αντικείμενο Γ

Πίνακας 1 Σχήμα Posting List

Για να περιοριστεί το μέγεθος της λίστας με τις συχνότητες εμφάνισης των λέξεων ταξινομούνται οι λέξεις κάθε αντικειμένου με αύξουσα σειρά των συχνοτήτων εμφάνισης τους. Τότε ορίζουμε ένα pprefix(x) το οποίο υπακούει στην εξίσωση: l_p^x=|x|-θ*|x|+1 . Η εξίσωση αυτή στην ουσία ορίζει από κάθε αντικείμενο x=(A,B,C,D) με μέγεθος |x|=4 και κατώφλι θ=0,7 να επιλεγούν οι 2 πρώτοι χαρακτήρες (A,B) οι οποίοι αποτελούν το pprefix(x).Αν τουλάχιστον ένας από τους χαρακτήρες (λέξεις) του pprefix(x) περιλαμβάνεται και στο pprefix() οποιουδήποτε άλλου αντικειμένου, τότε παράγεται υποψήφιο ζευγάρι 2 αντικειμένων. Σε βοηθητικό ρόλο του pprefix(x) δημιουργήθηκε και το iprefix(x) το όποιο έχει ένα ευρετήριο με τους όρους που εμπεριέχονται στο pprefix(x). Το μήκος του iprefix υπολογίζεται από τον εξής τύπο: l_i^x=|x|-|[(2*θ)/(1+θ)]*|x||+1

2.1.2 Spatial Objects

Για την δημιουργία ενός ευρετηρίου για δεδομένα που περιέχουν σημεία (x,y) με γεωγραφικές συντεταγμένες χρησιμοποιήθηκαν ποικίλες μεθοδολογίες όπως είναι ο διαμοιρασμός του χώρου βασιζόμενος σε διαφόρων ειδών δέντρα όπως KD-tree,Quad-tree,R-tree και ο διαμοιρασμός του χώρου σε ίσα τετράγωνα Grid. Στη συνέχεια περιγράφονται οι παραπάνω μεθοδολογίες.

R-tree: Σύμφωνα με το δέντρο R-tree δημιουργείται ένα ιεραρχικό δυαδικό δέντρο χωρίζοντας τον χώρο σε επιμέρους τμήματα (MBR minimum bound rectangle) μέχρι να φτάσει το κλαδί του δέντρου που αποτελεί το μικρότερο mbr με μέγιστη απόσταση e μεταξύ των σημείων που περιλαμβάνονται στο mbr. MBR είναι το σύνολο των δεδομένων και σε κάθε ανάπτυξη του δέντρου το ένα mbr θα χωριστεί σε δύο υποσύνολα δεδομένων.

KD-tree: Το KD-tree είναι δυαδικό δέντρο, το οποίο μπορεί να εφαρμοστεί και σε δεδομένα όπου το κάθε σημείο που περιέχεται στα δεδομένα αυτά αποτελείται από περισσότερες από δύο διαστάσεις x,y,z. Η μεθοδολογία ανάπτυξης του δέντρου διαφέρει από αυτή του R-tree. Στο KD-tree αναπτύσσεται ένας κόμβος για κάθε διάσταση των σημείων του δέντρου. Πιο συγκεκριμένα, στο πρώτο επίπεδο ανάπτυξης του δέντρου λαμβάνεται από όλα τα σημεία υπόψη η διάσταση x των σημείων και υπολογίζεται η διάμεσος αυτών. Σύμφωνα με την διάμεσο της διάστασης x δημιουργούνται οι δύο πρώτοι κόμβοι για τα σημεία που έχουν x>διάμεσου και για x<=διάμεσου. Στη για κάθε ένα από τα δύο υποσύνολα δεδομένων που έχουν δημιουργηθεί υπολογίζεται η διάμεσος για τη διάσταση y των σημείων και με των ίδιο τρόπο αναπτύσσεται ένα ακόμα επίπεδο του δέντρου όπου έχουν διαμοιραστεί τα σημεία σε τέσσερα πλεον υποσύνολα τώρα πλέον. Τέλος με ακριβώς τον ίδιο τρόπο αναπτύσσεται ένα ακόμα επίπεδο του δέντρου χρησιμοποιώντας την διάσταση z των σημείων.

Quad-tree: Το Quad-tree είναι ένα δέντρο όπου σε κάθε ανάπτυξή του παράγει τέσσερις κόμβους παιδιά με αποτέλεσμα να χωρίζει το επίπεδο σε τέσσερα ίδια σχήματα. Ο κάθε κόμβος του δέντρου έχει ένα άνω κατώφλι χωρητικότητας σημείων του δέντρου. Όταν ξεπεραστεί αυτό το κατώφλι, ο κόμβος διασπάται και αναπτύσσεται έναν ακόμα επίπεδο του δέντρου.

Grid: Η μεθοδολογία του Grid δεν βασίζεται στην μεθοδολογία των δέντρων αλλά με την χρήση της συγκεκριμένης μεθοδολογίας χωρίζεται ο χώρος σε ίσα τετράγωνα.

2.2 Partitioning

Για να επιτευχθεί ο διαμοιρασμός των δεδομένων και η παραλληλοποίηση του φόρτου εργασίας χρησιμοποιούνται διάφορες τεχνικές partitioning. Μια από αυτές είναι το grid που βασίζεται στα χωρικά γνωρίσματα των δεδομένων και χωρίζει τον χώρο που καλύπτουν τα δεδομένα σε ίσα τετράγωνα όπου

η πλευρά του τετραγώνου ισούται με το κατώφλι της απόστασης e που έχει δώσει ο χρήστης. Ορίζοντας ένα δυναμικό grid χωρίζεται ο χώρος σε έως 9 ίσα τετράγωνα όπου εξετάζονται τα σημεία που βρίσκονται στο κεντρικό κελί και αυτά που είναι στα γειτονικά του.

			44	45	46		
			36	37	38		
			28	29	30		
9	10	11					
1	2	3					

Πίνακας 2 Grid Partitioning

Ένας άλλος τρόπος διαμοιρασμού των δεδομένων είναι το KD-Tree partitioning που βασίζεται στα δέντρα KD-Tree, τα οποία είναι δυαδικά δέντρα που σε κάθε κόμβο χωρίζουν τον χώρο σε 2 ίσες περιοχές.

2.3 Grouping Objects

Για την καλύτερη απόδοση των αλγορίθμων χωροκειμενικής σύζευξης εφαρμόστηκαν τεχνικές ομαδοποίησης αντικειμένων ανάλογα με το τι κάθε αλγόριθμος εξετάζει πρώτα, κείμενο ή χώρο.

Όσο αφορά τους αλγορίθμους που εξετάζουν πρώτα την ομοιότητα κειμένου, ομαδοποιήθηκαν ανάλογα με το prefix των αντικειμένων όλα τα αντικείμενα με το ίδιο prefix και κατατάχθηκαν μαζί. Μια επιπλέον παράμετρος ομαδοποίησης που μπορεί να συσχετιστεί μαζί με το prefix, είναι και το μέγεθος των αντικειμένων.

Για τους αλγορίθμους που εφαρμόζουν κάποια τεχνική partitioning είτε με grid είτε με R-tree εφαρμόζοντας την μέθοδο της ομαδοποίησης, ομαδοποιούνται πρώτα τα αντικείμενα με βάση το prefix και κατά δεύτερον βάσει του μεγέθους τους, και στην συνέχεια εφαρμόζεται η τεχνική partitioning. Αυτό οδηγεί στο να δημιουργηθεί μεγάλος αριθμός από ομάδες, αλλά παρόλα αυτά στους αλγορίθμους που εφαρμόζουν ομαδοποίηση αντικειμένων οι Bouros et.al σημειώνουν βελτίωσή στις περισσότερες περιπτώσεις.

2.4 Similarity join query

Για να επιβεβαιώσουμε αν ένα υποψήφιο ζευγάρι από 2 αντικείμενα έχει την απαιτούμενη ομοιότητα στο κείμενο και στην απόσταση των σημείων μεταξύ τους, χρησιμοποιούμε τις εξής μετρικές για 2 σημεία x,y.

- Ευκλείδεια απόσταση: dist(x,y)= $\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$ για την χωρική απόσταση των σημείων μεταξύ τους
- Ομοιότητα Jaccard: sim(x,y)= |(x_i.text ⋂ y_i.text)|/(|x_i.text|+|y_i.text|-|(x_i.text⋂ y_i.text)|) όπου x.text και y.text είναι οι λέξεις που έχουν τα αντικείμενα x,y

Στο paper των Bouros et al. αναφέρονται 2 τεχνικές που βασίζονται στις παραπάνω μετρικές οι: SPSJOIN,STSJOIN.

- Η SPSJOIN συνδυάζει και τις 2 μετρικές σε μία, σύμφωνα με τον εξής τύπο:sim(x,y)= sim_t(x,y)/(1+dist(x,y)), ενώ το STSJOIN έχει 2 ανεξάρτητες μετρικές, μια για την απόσταση και μια για την κειμενική ομοιότητα
- Το SPSJOIN για κάθε αντικείμενο x επιστρέφει το αντικείμενο y με την μεγαλύτερη ομοιότητα, ενώ το STSJOIN επιστρέφει όλα τα ζευγάρια x,y
- Το SPSJOIN μπορεί να επιστρέψει και πλεονάζουσα πληροφορία

2.5 Αλγόριθμοι

2.5.1 All-Pairs

Ο αλγόριθμος All-Pairs χρησιμοποιώντας της λογική του prefix έχει παράγει υποψήφια ζευγάρια από αντικείμενα βάσει των λέξεων που εμπεριέχονται στο prefix κάθε αντικειμένου. Οπότε για την επιβεβαίωση της ομοιότητας 2 αντικειμένων πρέπει να εξεταστούν και οι λέξεις που δεν περιέχονται στο prefix οι οποίες αποτελούν το suffix κάθε αντικειμένου. Για να υπολογίσουμε το μέγιστο overlap O(x,y) των suffixes θα χρησιμοποιήσουμε τον εξής τύπο: O(x,y)=1+min(|suff(x)|,|suff(y)|). Πλέον χρησιμοποιώντας την ομοιότητα Jaccard υπολογίζεται η ομοιότητα των ζευγαριών (x,y)

2.5.2 PPJOIN

Ο αλγόριθμος PPJOIN είναι μια επέκταση το All-Pairs αλγορίθμου χρησιμοποιώντας φίλτρο κατάληξης και φίλτρο θέσης.

- Χρησιμοποιεί φίλτρο για να παράγει υποψήφια ζευγάρια χρησιμοποιώντας την λογική του pprefix
- Αφού παράγει τα υποψήφια ζευγάρια χρησιμοποιεί ένα φίλτρο μεγέθους για 2 αντικείμενα (x,y) όπου |y|>=θ*|x| και ένα φίλτρο θέσεως όπου ελέγχει αν το άνω όριο της επικάλυψης των κοινών όρων του x,y O(x,y) είναι μικρότερο από την επικάλυψη του x και του y. Τότε τα ζευγάρια (x,y) είναι ασυσχέτιστα.
- Στο επόμενο φίλτρο ελέγχονται οι εναπομείναντες λέξεις του κάθε αντικειμένου με ένα φίλτρο θέσεως και κατάληξης αυξάνοντας μετά τον έλεγχο κάθε λέξεις το overlap κατά 1 που στην ουσία είναι ο δείκτης για τις λέξεις που ταιριάζουν.

2.5.3 PPJ

Ο PPJ είναι μια επέκταση του PPJOIN αλγορίθμου χρησιμοποιώντας ένα επιπλέον φίλτρο για την απόσταση των αντικειμένων (x,y) αφότου αυτά έχουν περάσει από την διαδικασία του prefix φιλτραρίσματος. Δηλαδή τα φίλτρα έχουν την εξής σειρά:

- Prefix Filtering
- Suffix Filtering και Distance Filtering
- Positional Filtering και Suffix Filtering

2.5.4 PPJ-I

Ο PPJ-I είναι επέκταση του αλγορίθμου PPJ χρησιμοποιώντας το grid partitioning. Κάνει τα εξής βήματα:

- Για κάθε κελί c ανακαλύπτει περιμετρικά από αυτό έως 9 κελιά που αποτελούν έως 3 διαστήματα κελιών

- Για κάθε εγγραφή υπάρχουν 2 δείκτες ένας εκ των οποίων ορίζει το id του κελιού cid και ο άλλος ορίζει τα αντικείμενα που έχει κάθε κελί pcid. Χρησιμοποιώντας το pcid και την posting list Lt έχουμε την posting list κάθε αντικειμένου Lt.pcid
- Μετά τις παραμετροποιήσεις για την υποστήριξη του grid, οι υπόλοιπες τεχνικές που εφαρμόζει ο αλγόριθμος είναι ίδιες με του PPJ.

2.5.5 PPJ-C

Ο PPJ-C βασίζεται στον PPJ αλγόριθμο και είναι μια παραλλαγή του PPJ-I. Ο PPJ-C ακολουθεί τα εξής βήματα για την υλοποίηση του:

Αφού χωριστεί ο χώρος με την βοήθεια του grid ο αλγόριθμος εξετάζει τα κελιά με αύξουσα σειρά του cell.id

- Χρησιμοποιώντας την τεχνική του grid χωρίζει τον χώρο με την ίδια λογική με τον PPJ-I και κρατά ένα cell.id για κάθε κελί. Βέβαια κατά την διάρκεια της εύρεσης υποψήφιων ζευγαριών ο αλγόριθμος εξετάζει το κεντρικό κελί και τα γειτονικά κελιά που έχουν μικρότερο ids από το κεντρικό κελί.
- Ένα επιπλέον προτέρημα του PPJ-C είναι ότι για κάθε κελί c δημιουργεί μια posting list όποτε υπάρχει μεγάλη εξοικονόμηση χώρου διότι για κάθε έλεγχο που κάνει σε γειτονικά κελιά c’ κατά την διαδικασία του prefix filtering ελέγχεται η posting list του c (c.Lt) και του c’ (c’.Lt) και παράγονται κατευθείαν τα υποψήφια ζευγάρια με περιορισμένου όγκου posting lists.
- Επίσης στον PPJ-C ελέγχουμε τα αντικείμενα του κεντρικού κελιού c μήπως υπάρχει κάποια σύζευξη. Κατά του ελέγχουμε των αντικειμένων του κελιού c εκμεταλλευόμενη την ιδιότητα του PPJ να ελέγχει τα αντικείμενα σε αύξοντα σειρά μεγέθους έτσι πράττεται και PPJ-C. Προχωρώντας στον έλεγχο των γειτονικών κελιών c’ τα αντικείμενα των 2 κελιών ενοποιούνται και ταξινομούνται κατά αύξουσα σειρά μεγέθους.

2.5.6 PPJ-R

Ο PPJ-R είναι μια επέκταση του PPJ όπου τα αντικείμενα έχουν υποστεί χωρικό indexing με την βοήθεια των R-trees. Αφού ο αλγόριθμος έχει ένα χωρικό κατώφλι απόστασης e ο αλγόριθμος παίρνει σαν είσοδο 2 κόμβους Nx ,Ny αν Nx=Ny που αποτελούν τη ρίζα του δέντρου. Αν οι Nx, Ny είναι μη επικαλυπτόμενοι κόμβοι τότε ανακαλύπτει ζευγάρια των οποίων η απόσταση τους είναι ίση με το κατώφλι που έχουμε ορίσει.

Διαφορές PPJ-C,PPJ-R:

- Το χωρικό partitioning ορίζεται από την δομή του R-tree και όχι από το δυναμικό grid partitioning
- Ο R-Tree χρησιμοποιείται για να βρίσκει ζευγάρια από χωρικά δεδομένα και απαιτείται για την ένωση η χρήση του PPJ

Από το paper των Bouros et al. και τον αλγορίθμο PPJ-C που παρεμπιπτόντως είχε και τις καλύτερες επιδόσεις είναι αξιοσημείωτη η τεχνική του indexing και partitioning που χρησιμοποιεί: Θα μπορούσαμε να εκμεταλλευτούμε το γεγονός ότι χρησιμοποιεί posting list για κάθε κελί του grid που έχει ορίσει. Κρατώντας την posting list για κάθε χωρίο του χώρου μπορούμε να εφαρμόσουμε μια υβριδική στρατηγική grid η οποία να χωρίζει το χώρο σύμφωνα με ένα κατώφλι e αλλά και με την πυκνότητα των

σημείων ανά χωρίο. Επίσης νομίζω ότι θα ήταν αποδοτικότερο να διασφαλίζεται ότι τα αντικείμενα μπορούν να σχηματίσουν ζευγάρια από την πλευρά της χωρικής απόστασης και μετά να προβαίνει κανείς στον οποιοδήποτε έλεγχο κειμενικής ομοιότητας. Πιο εκτεταμένη ανάπτυξη αυτής της εφαρμογής γίνεται στην τελευταία ενότητα "Υλοποίηση για περαιτέρω διερεύνηση".

Στον επόμενο πίνακα συγκρίνονται οι αλγόριθμοι που αναφέρθηκαν στο paper των Bouros et.al και επιλύουν του ερώτημα του Spatio-textual join σε κεντρικοποιημένο περιβάλλον.

Algorithms	**Partitioning-Filter-Join**	**Partitioning Index**	**Spatial Distance**	**Textual Similarity**
All-Pairs	-	-	-	Jaccard
PPJ	-	-	Euclidean	Jaccard
PPJ-I	Spatial-Textual-Spatial	Grid	Euclidean	Jaccard
PPJ-C	Spatial-Textual-Spatial	Grid	Euclidean	Jaccard
PPJ-R	Spatial-Spatial-Textual	R-Tree	Euclidean	Jaccard

Πίνακας 3 Κεντρικοποιημένοι Αλγόριθμοι

2.5.7 Spatio-Textual Join in Distributed Environment

Με βάση τους αλγορίθμους που αναφέρθηκαν για την χωροκειμενική σύζευξη σε κεντρικοποιημένο περιβάλλον. Προχωρούμε στην ανάλυση και παραμετροποίηση αυτών για την λειτουργεία τους σε κατανεμημένο σύστημα. Σύμφωνα με την παραπομπή[2]. Για την δημιουργία αλγορίθμου χωροκειμενικής σύζευξης ο οποίος να λειτουργεί σε κατανεμημένο περιβάλλον με την βοήθεια της map-reduce. Βάση αποτέλεσε ο αλγόριθμος PPJ-C για κεντρικοποιημένο περιβάλλον. Οι διαδικασίες που ακολουθήθηκαν από των Zhang et.al είναι οι εξής:

Για την προ επεξεργασία των δεδομένων, με σκοπό να ταξινομηθούν οι λέξεις των αντικειμένων με αύξουσα σειρά σύμφωνα με την συχνότητα εμφάνισής τους, χρειάστηκαν 2 map-reduce. Στην πρώτη map-reduce η συνάρτηση map χωρίζει σε λέξεις το κείμενο που περιλαμβάνει κάθε αντικείμενο και στην reduce για κάθε αντικείμενο υπολογίζεται ο αριθμός των λέξεων που επαναλαμβάνεται μέσα σε αυτό και παράγεται ένα αποτέλεσμα της μορφής (λέξεις, μετρητής). Στην δεύτερη map-reduce στην map συνάρτηση εισέρχονται τα ζευγάρια (λέξεις, μετρητής) και ταξινομούνται με αύξουσα σειρά ανάλογα με την συχνότητα εμφάνισής τους. Αυτή τεχνική ονομάζετε Global Token Ordering (GTO). Επίσης αν έχουν υποστεί grid τα δεδομένα χρησιμοποιείται ακριβώς η τεχνική διαχείρισής τους όπως στον αλγόριθμο PPJ-C και αυτή ονομάζεται για το συγκεκριμένο paper, Grid Cell Assignment Principle(GCA).

Για ένα πλήρη αλγόριθμο που βασίζεται στο δυναμικό grid partitioning εκτελούνται τα εξής βήματα:

- Συνάρτηση όπου λαμβάνονται τα όρια του γεωγραφικού χώρου που καλύπτουν τα δεδομένα
- Εφαρμόζεται GTO
- Διαβάζονται τα αντικείμενα και υπολογίζονται τα prefix κάθε αντικειμένου
- Εφαρμόζεται grid partitioning. Το cell-id δηλώνει τον αριθμό του κελιού
- Ομαδοποίηση των αντικειμένων σύμφωνα με το GCA

- Κάθε παραγόμενο υποψήφιο ζευγάρι έχει ένα σύνθετο κλειδί που αποτελείται από το:group id και μέγεθος του αντικειμένου. Χρησιμοποιώντας αυτές τις τεχνικές partitioning επιτεύχθηκε τα αντικείμενα να είναι ομαδοποιημένα ανά group id και ταξινομημένα σε αύξουσα σειρά group id και μεγέθους
- Όλα τα αντικείμενα στέλνονται στον ίδιο reducer και εξετάζονται ανά group εφαρμόζοντας φίλτρο για την απόσταση των ζευγαριών.
- Κατά την διάρκεια της σύζευξης εφαρμόζονται 2 ειδών inverted index, ένας για το κεντρικό κελί κάθε grid και ένας για την συσχέτιση με τα γειτονικά κελιά.

Για την βελτιστοποίηση της απόδοσης του αλγορίθμου πρέπει να υπολογιστεί ο αριθμός των duplicates που μεταφέρονται μέσω δικτύου χρησιμοποιείται ένα cprefix που είναι ή ένωση όλων των prefix των αντικειμένων που υπάρχουν σε ένα κελί του grid.Τα αντικείμενα με το ίδιο cprefix στέλνονται στον ίδιο reducer οπότε πριν την reduce έχουν γίνει όσα duplicates χρειάζονται.

Μια άλλη προσέγγιση στον διαμοιρασμού του φόρτου εργασίας στους υπολογιστές είναι η χρήση των KD-Trees. Σύμφωνα με τη προσέγγιση αυτή, αντί να χωριστεί ο χώρος σε κελιά ίσου χωρικού μεγέθους χωρίζεται σε μικρά KD-Tree ανάλογα με την κατανομή των αντικειμένων που έχει κάθε περιοχή.

Παρακάτω παρατίθενται οι αλγόριθμοι που χρησιμοποιήσαν μερικές από τις προηγούμενες τεχνικές.

Algorithms	**Partitioning-Filter-Join**	**Group**	**Partitioning Index**	**Spatial Distance**	**Textual Similarity**
PPJOIN+	-		-	-	Jaccard
STSJ-B	Spatial-Textual-Spatial	Prefix after Grid	Grid	Euclidean	Jaccard
STSJ-C	Spatial-Textual-Spatial	Cprefix in Cell of Grid	Grid	Euclidean	Jaccard
STSJ-K	Spatial-Textual-Spatial	Groups With Locations	KD-Tree	Euclidean	Jaccard

Πίνακας 4 Κατανεμημένοι Αλγόριθμοι

Ο αλγόριθμος βελτιώθηκε αρχικά με την χρήση των cprefix για όλα τα κελιά του grid και κατά δεύτερον με την χρήση το KD-Tree partitioning που είναι καλύτερο από το grid. Οι δύο μέθοδοι που προαναφέραμε βοηθούν στην μείωση των duplicates και στην εξισορρόπηση του φόρτου εργασίας στους reducer στην φάση της σύζευξης. Παρατηρούμε από τις δοκιμές που έγιναν ότι οι αλγόριθμοι που βελτιώθηκαν σημειώνουν καλύτερους χρόνους από τον αρχικό.

2.5.8 Top-K

Μια διαφορετική προσέγγιση στο χωροκειμενική σύζευξη δεδομένων είναι η εύρεση ζευγαριών από αντικείμενα που έχουν την μικρότερη απόσταση μεταξύ τους και τον μεγαλύτερο βαθμό ομοιότητας κειμένου.

Σύμφωνα με την βιβλιογραφία [4] που αναφέρεται στο paper των Ballesteros et al. όπου υπολογίζει τα ζευγάρια με την μικρότερη απόσταση και μεγαλύτερη κειμενική ομοιότητα σε cluster υπολογιστών χρησιμοποιώντας την μέθοδο της MapReduce και spatial partitioning. Για το partitioning δημιουργήθηκε

χρησιμοποιήθηκε ο αλγόριθμος συσταδοποίησης X-means όπου κάθε συστάδα αποτελούσε ένα partition.

Για εύρεση των καλύτερων ζευγαριών δύο αντικειμένων (r,s) ανά partition χρησιμοποιήθηκε η εξής συνάρτηση ομοιότητας: sim(r,s)=$sim_t(a_r,a_s)$/(1+dist(p_r, p_s))

Όπου:

- $sim_t(a_r,a_s)$ είναι η συνάρτηση που υπολογίζει την ομοιότητα του κειμένου μεταξύ των αντικειμένων (r,s) και εφαρμόζει την συνάρτηση Jaccard,
- dist(p_r, p_s) είναι η συνάρτηση που υπολογίζει την γεωγραφική απόσταση των αντικειμένων (r,s) και εφαρμόζει την συνάρτηση της απόσταση Great Circle[1]

Έχοντας τα δεδομένα σε partition γίνονται 2 επαναλήψεις, μια για να βρεθεί ο βαθμός ομοιότητας κάθε ζευγαριού σύμφωνα με την παραπάνω σχέση και τα ζευγάρια που έχουν τον μεγαλύτερο βαθμό ομοιότητας κρατούνται για κάθε partition. Επίσης στην πρώτη επανάληψη γίνεται έλεγχος για αντικείμενα που βρίσκονται εκατέρωθεν στα όρια δύο συστάδων και η απόστασή τους να είναι μικρή. Στην δεύτερη επανάληψη βρίσκονται τα καλύτερα από τα ζευγάρια που δημιουργούνται από αντικείμενα που βρίσκονται σε διαφορετικούς cluster. Στην τρίτη επανάληψη βρίσκονται τα ζευγάρια που έχουν την μεγαλύτερη ομοιότητα χωρική και κειμενική από όλους τους cluster.

Επίσης μια ακόμα προσέγγισή που αναφέρεται στο paper των Jinfeng Rao at el.[5] χρησιμοποιεί για την χωροκειμενική σύζευξη των αντικειμένων (r,s) την cosine similarity αντί της Jaccard για την κειμενική ομοιότητα και την Euclidean απόσταση για την απόσταση μεταξύ των αντικειμένων (r,s). Για το partitioning των δεδομένων χρησιμοποιήθηκε quad tree και για το index τον δεδομένων χρησιμοποιήθηκε index με βάση το χώρο όπου πάνω σε αυτά τα δεδομένα έτρεξε αλγόριθμος που βασίζεται στον All-Pairs. Για τον τρόπο του index υπάρχουν 2 προσεγγίσεις οι:local index και η global index.

- Local Index: Σε αυτή την προσέγγιση συνδυάζεται ένα quad-tree με τον All-Pairs. Πιο συγκεκριμένα, δημιουργήθηκε ένα PR-quad-tree σε όλο το dataset, με βάση τις συντεταγμένες των αντικειμένων. Αφού αναπτύχθηκε το PR-quad-tree με τον περιορισμό ότι κάθε κελί θα είναι μικρότερο είτε ίσο από το κατώφλι της απόστασης που έχει ορίσει ο χρήστης. Για την εύρεση πιθανόν ζευγαριών με έναν αντικείμενο που βρίσκεται σε έναν κελί k γίνεται αναζήτηση αντικειμένων σε όλα τα κελιά του δέντρου γύρω από το κελί k. Επίσης δημιουργείται inverted index τοπικά για τα κελιά στα οποία γίνεται η αναζήτηση των πιθανών ζευγαριών.
- Global Index: Επίσης συνδυάζεται ένα quad-tree με τον All-Pairs με την μόνη διαφορά ότι δημιουργείται ένας global inverted index και κατά τη διάρκεια του spatial partitioning η κάθε posting list μεταφέρεται στο ανάλογο partition και κάθε posting list ταξινομείται με βάση την z-order. Οπότε, για κάθε αντικείμενο σε ένα κελί k ελέγχονται τα γύρω κελιά με βάση την z-order και όχι απλά τα γειτονικά κελιά. Επίσης, με την βοήθεια δεικτών γίνεται μια παραμετροποίησή στην z-order ώστε να μπορεί να εφαρμοστεί και σε grid.

[1] https://en.wikipedia.org/wiki/Great-circle_distance

Μέσα από τις δοκιμές των Jinfeng Rao at el. παρατηρούμε ότι σε μερικές περιπτώσεις ξεχωρίσει ελαφρώς κατά το indexing time και το running time οι αλγόριθμοι που είχαν global grid και global quad tree.

Για την εύρεση των top-k ζευγαριών στο συγκεκριμένο paper των Huiqi Hu et al.[3] ακολουθούνται τα εξής βήματα:

- **Textual Similarity**: μέσω της ομοιότητας Jaccard
- **Spatial Similarity**: για 2 αντικείμενα (r,s) είναι η εξής sim(r,s)=max(0.1-DIST(r,s)/DISTmax) όπου DIST() είναι η απόσταση μεταξύ των αντικειμένων (r,s) και DISTmax είναι η μέγιστη απόσταση που έχει ορίσει ο χρήστης
- **Spatio-Textual Similarity**: Η χωροκειμενική ομοιότητα υπολογίζεται μέσω του τύπου: sim_{st}(r,s)=a*sim_t(r,s)+(1-a)*sim_s(r,s) όπου sim_t(r,s) είναι η ομοιότητα του κειμένου των 2 αντικειμένων και sim_s(r,s) είναι η απόσταση των δύο αντικειμένων και a είναι η παράμετρος που καθορίζει την συσχέτιση μεταξύ ομοιότητας και απόστασης.
- **Top-K Spatio-Textual Similarity Join**: επιστρέφει ένα σύνολο ζευγαριών που έχουν τον μεγαλύτερο δείκτη ομοιότητας sim_{st}().

Για κάθε εγγραφή δημιουργείται ένα signature sig(r|tk)={[t,n]} όπου περιέχει r είναι η εγγραφή, tk είναι το κατώφλι ομοιότητας,t είναι οι όροι της εγγραφής και n είναι οι συντρεγμένες της εγγραφής. Με τους όρους των signatures δημιουργείται ένας inverted index όπου η είσοδος του είναι signature και κάθε signature συσχετίζεται με μια λίστα από εγγραφές που περιέχουν τα signatures.Τα ζευγάρια των εγγραφών που έχουν μικρότερη ομοιότητα από το κατώφλι απορρίπτονται. Για το partitioning των δεδομένων χρησιμοποιείται quad tree.

Για την δημιουργία του αλγορίθμου γίνονται τα εξής βήματα:

- Αφού δημιουργείται ένα spatial index
- Επιλέγονται k ζευγάρια στον ίδιο κόμβο παιδί και μπαίνουν σε μια ουρά προτεραιότητας
- Οι όροι κάθε εγγραφής ταξινομούνται με αύξουσα συχνότητα εμφανίσεις των όρων σε όλες τις εγγραφές.
- Ανακαλύπτεται για κάθε εγγραφή ο κόμβος στον οποίο περιέχεται.
- Για κάθε κόμβο n υπολογίζονται οι pivot terms.
- Για κάθε pivot term ανακτάται η αντίστοιχη posting list
- Αν η ομοιότητα των 2 αντικειμένων είναι μεγαλύτερη από το ορισμένο κατώφλι ενημερώνει την ουρά προτεραιότητας με αυτή την εγγραφή

Για την πρόσβαση στις εγγραφές των δεδομένων παρουσιάζονται 3 διαφορετικές τρόποι για την υλοποίηση τους τα signature αναπαρίστανται ως εξής (r,t,n) όπου r είναι η εγγραφή και (t,n) είναι η signature του r. Οι υλοποιήσεις είναι οι εξής:

- Textual First: Οι εγγραφές ταξινομούνται ανάλογα με το t
- Spatial First: Οι εγγραφές ταξινομούνται ανάλογα με τον κόμβο n
- Best First: Οι εγγραφές ταξινομούνται ανάλογα με το score κάθε τριπλούς
- Order-Aware Pruning: Αν των score κάθε τριπλέτας είναι μικρότερο από το ορισμένο κατώφλι η τριπλέτα αγνοείται.

Μετά από πειράματα προτείνεται από τους Huiqi Hu et al. η υλοποίηση με που βασίζεται στο καλύτερο similarity “Best First” έχεις τις καλύτερες επιδόσεις.

Στο paper των Tsatsanifos et al.[6] παρουσιάζεται ένα σύστημα query και top-k απαντήσεων σε αυτό το query με βάση της χωροκειμενικές τους ιδιότητες για να εξαχθεί η top-k απάντηση κάνουμε τα εξής βήματα:

Preference score s(t) of feature object: όπου s(t)=(1-λ)*t.s+λ*sim(t,W). Το t.s δηλώνει το non-spatial score του t όπου το t μπορεί να είναι η βαθμολογία των αξιολογήσεων του feature object και sim(t,W) είναι η ομοιότητα του κειμένου μεταξύ του query αναζήτησης και query απάντησης για το feature object που υπολογίζεται μέσω της Jaccard. Και το λ είναι μια σταθερά μεταξύ του (0,1) όπου δηλώνει ο χρήστης τι τον ενδιαφέρει περισσότερο η κειμενική ομοιότητα ή η βαθμολογία του feature object. Όπου t είναι τα data object δηλαδή τα σημεία του χώρου που χρησιμοποιούνται σαν σημεία αναφοράς για την εύρεση άλλων σημείων με τα βάση τα data object τα οποία ονομάζονται feature object.

Preference score τ_i(p) of data object: όπου τ_i=max{s(t)|t∈ F_i:dist(p,t)<r and sim(t,W_i)>0} το σύνολο των feature object αποτελεί ένα feature set (F_i). Το dist(p,t) είναι η απόσταση μεταξύ ενός data object(p) και ενός feature object(t) που μετράται με την Euclidean απόσταση.Το sim(t,W_i)>0 η ομοιότητα του μεταξύ του query αναζήτησης για query απάντησης για το data object.

Spatio-textual preference score of data object τ(p): όπου τ(p)=$\sum_{i=1}^{c} t_i(p)$ με αυτή την εξίσωση υπάρχει η δυνατότητα της εύρεσης feature object με διαφορετική θεματική ενότητα γύρω από ένα σημείο αναφοράς. Για παράδειγμα feature object που το ένα μπορεί να είναι καφετέρια και το άλλο εστιατόριο.

1.1.1 Indexing

Στα data object έγινε index με βάση το r-tree επίσης και feature ακολουθούν την ίδια λογική του r-tree index. Επίσης χρησιμοποιείται και ένας τρόπος indexing που βασίζεται στην Hilbert Curve για ένα σύνολο από tokens που γίνονται αναζήτηση στα tokens των feature object. Για τον αριθμό τον tokens της αναζήτησης δημιουργείται ένας vector που αποτελείται από τόσα bit όσα είναι τα tokens και όταν υπάρχει κοινό token μεταξύ της αναζήτησης και των feature object το bit γίνεται ένα. Αυτό το vector λέγεται Hilbert value και η απόσταση μεταξύ των διαφορετικών Hilbert values υπολογίζεται με μια αφαίρεση. Οπότε κάθε σημείο έχει 4 γνωρίσματα (t.x,t.y,t.s,H(t.W)) και αυτός ο τρόπος index ονομάζεται SRT-index.

Μια άλλη μέθοδος προσπέλασης των δεδομένων ο υπολογισμός του Spatio-textual score t(p) για κάθε data object και η επιστροφή των top-k data object με το μεγαλύτερο score και αυτό ονομάζεται STDS.

Μια άλλη στρατηγική φόρτωσής των Spatio-textual δεδομένων είναι το STPS όπου κάνει την εξής διαδικασία βρίσκει τα feature object με το μεγαλύτερο score και επιστρέφει τα data object που είναι κοντά με τα συγκεκριμένα feature object.

Ένας διαφορετικός τύπος query είναι influence preference score τ_i(p) όπου τ_i(p)=max{s(t)*$2^{-dist(p,t)/r}$|t∈ F_i:sim(t,W_i)>0} όπου δηλώνεται το συνολικό Spatio-textual score και επιστρέφονται τα top-k data object με το μεγαλύτερο score.

Τέλος δημιουργήθηκαν και τα STPQ query όπου nearest neighbor preference score τ_i(p)={s(t)|t∈ F_i:dist(p,t)<=dist(p,t')∀t' ∈ F_i and sim(t,W_i)>0}. Στο συγκεκριμένο query δηλώνεται ένας εύρος από score τ(p) και επιστρέφονται τα top-k data object που βρίσκονται μέσα σε αυτό το εύρος.

3 Κεφάλαιο - Αλγόριθμος

Κατά την διάρκεια της πειραματικής διαδικασία δημιουργήθηκε ένας αλγόριθμος για την χωρο-κειμενική σύζευξη δεδομένων που αποτελούνταν από 2 dataset αποτελούνται από συντεταγμένες (lon,lat), το όνομα(name) κάθε αντικειμένου που απεικονίζουν οι συντεταγμένες, την κειμενική πληροφορία κάθε αντικειμένου και την ετικέτα κάθε εγγραφής. Συνοψίζοντας το σχήμα των δεδομένων είναι το εξής:

Name	Text
Lon	Float
Lat	Float
Textual Field	Text
Label	Keyword

Πίνακας 5 Σχήμα πρωτότυπου Συνόλου Δεδομένων

Σκοπός της υλοποίησης είναι η εύρεση ζευγαριών αντικειμένων με διαφορετική ετικέτα όπου η απόσταση τους υπακούει σε μια προκαθορισμένη απόστασή R και η κειμενική ομοιότητα τους υπακούει σε έναν βαθμό ομοιότητας K ελεγχόμενο μέσω της ομοιότητας των δύο κειμένων που μετράται με την απόσταση Jaccard.

3.1 Partitioning

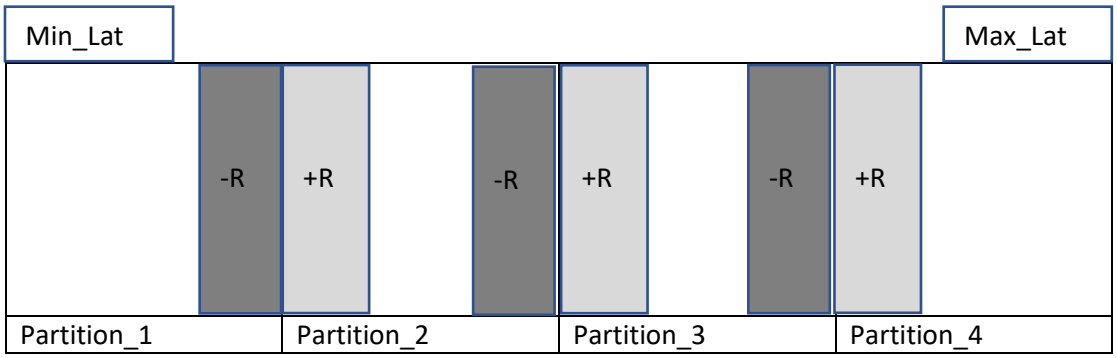
Στην παρούσα περίπτωση τα δύο σύνολα δεδομένων που χρησιμοποιήθηκαν αποτελούνταν από εστιατόρια και ξενοδοχεία. Με το εξής σχήμα δεδομένων:

Name	Text
Lon	Float
Lat	Float
Facilities	Text
Label	Keyword

Πίνακας 6 Σχήμα Συνόλου Δεδομένων

Κατά την διάρκεια του partitioning χρησιμοποιήθηκε μια τεχνική που χώριζε τον όγκο των δεδομένων με βάση την συντεταγμένη lat. Πιο συγκεκριμένα διαβάζονται τα δυο dataset restaurant(name,lan,lot,Facilities) και hotel(name,lan,lot,Facilities) αφού προστεθεί μια ετικέτα (Label) R για τα εστιατόρια και H για τα ξενοδοχεία τα δύο dataset ενοποιούνται. Στην συνέχεια ακολουθεί το partitioning θέτοντας N-1 περιορισμούς στο lat ώστε να έχουμε N τμήματα του ενοποιημένου dataset που περιέχουν παραπλήσιο αριθμό εγγραφών για να επιτύχουμε πλήρη εξισορρόπηση στους 2 slaves του δικτύου μας. Παρατηρήθηκε ότι με τον διαχωρισμό αυτό είχαμε απώλειες σε πιθανά ζευγάρια λόγω του ότι κοντά στο σημείο τομής lat μπορεί να ένα ζευγάρι να ήταν εκατέρωθεν του σημείου τομής. Δηλαδή ένα ξενοδοχείο να είναι στο partition 1 και ένα εστιατόριο να είναι στο partition 2 αλλά η

απόσταση μεταξύ τους να είναι μικρότερη ή ίση της απόστασης R.Οπότε για την αντιμετώπιση αυτού του προβλήματος έπρεπε για κάθε partition να αντιγράψουμε από τα υπόλοιπα partition τα εστιατόρια ή τα ξενοδοχεία που απαιτούνται για μην υπάρχει έλλειψη πιθανόν ζευγαριών (ξενοδοχείου, εστιατορίου). Επιλέξαμε τα ξενοδοχεία να είναι αυτά που θα αντιγράφονται λόγω του ότι έχουν σημαντικά λιγότερες εγγραφές τα εστιατόρια είναι 78.891 και τα ξενοδοχεία είναι 25.463.Για την υλοποίησης της αντιγραφής χρησιμοποιήθηκε μια δομή επανάληψης που επαναλαμβάνεται N-1 φορές και ελέγχει όλα τα σημεία τομής μεταξύ των partitions.



Στο παραπάνω σχήμα φαίνεται ο τρόπος που μεταφέρονται τα hotel από partition σε partition δηλαδή δημιουργούνται duplicates. Για την δημιουργία των duplicates γίνεται έλεγχος σε κάθε σημείο τομής με σειρά από τα αριστερά προς τα δεξιά, από την μικρότερη συντεταγμένη min_lat στην μεγαλύτερη max_lat. Στο σημείο τομής του partition_1 με το partition_2 στο ανοιχτό γκρι ορθογώνιο φαίνονται τα hotels που είναι στο partition_2 και πρέπει να πάνε στο partition_1. Το κριτήριο αντιγραφής κάθε hotel είναι: Lat_hotel – Lat_tomis >R, όπου Lat_tomis είναι το lat του σημείου τομής των 2 partitions, Lat_hotel είναι το lat του hotel που βρίσκεται στο partition_2 και R είναι η απόσταση που έχει ορίσει ο χρήστης. Εξίσου το ίδιο γίνεται για το σκούρο γκρι όπου περιέχει τις εγγραφές που είναι το partition_1 και πρέπει να αντιγραφούν στο partition_2 με την εξής διαφορά στη συνθήκη:Lat_tomis-Lat_hotel>R. Για να γνωρίζουμε σε ποιο partition είναι κάθε εγγραφή έχουμε δημιουργήσει μια μεταβλητή που δείχνει τον αριθμό του partition κάθε εγγραφής και με βάση αυτή την μεταβλητή δημιουργείται ο αριθμός των partitions.

3.2 Αλγόριθμος

Στον αλγόριθμο που δημιουργήθηκε αφού εκτελεστεί το διάβασμα των αρχείων στην συνέχεια εκτελείται η προαναφερθείσα διαδικασία του partitioning. Τέλος αφού τα δεδομένα είναι στην επιθυμητή μορφή και είναι διαχωρισμένα στα επιμέρους partition εκτελείται η διαδικασία του cross join ανά partition και υπολογίζεται ανά partition η χωρική απόσταση κάθε hotel από κάθε restaurant όπως επίσης υπολογίζεται και η κειμενική ομοιότητα μεταξύ των hotel ,restaurant για την κειμενική πληροφορία έχουν τα δεδομένα. Παρακάτω παρατίθεται ο ψευδοκώδικας όλου του αλγορίθμου.

```
1.  partiotion_number=8
2.  restaurant=read(file)
3.  hotel=read(file)
4.  merge_df=hotel.union(restaurant)
5.  max_lat=merge_df("Lat").max()
6.  min_lat=merge_df("Lat").min()
7.  partition=(max_lat-min_lat)/partition_number
8.  merge_df=merge_df.withcolumn(partition_label for each record)
9.  for each i in partition_number:
10.    hotel_min=merge_df.select(*).where(lat>max_lat – (partition_number – i) * partition) – R)
11.    AND lat< maxl_lat-(partition_number – i ) * partition)))
12.    hotel_max=merge_df.select(*).where(lat<max_lat – (partition_number - i) * partition) + R)
13.    AND lat> maxl_lat-(partition_number - i) * partition)))
14. partition_data = merge_df.union(hotel_min).union(hotel_max)
15. partition_data = partition_data.repartition(partition_label)
16. partition_data = partition_data(restaurant cross_join hotel by partition)
17. for each i in partition_data
18.       haversine_result = haversine(hotel, restaurant)
19.       jaccard_result = jaccard(hotel, restaurant)
```

Η επεξήγηση του αλγορίθμου θα γίνει βάση των αριθμού του γραμμών του ψευδοκώδικα.

- (1-1) Ο χρήστης έχει την δυνατότητα να ορίσει τον αριθμό τον partition που επιθυμεί να διχοτομηθούν τα δεδομένα
- (2-4) Φορτώνονται τα δύο dataset: restaurant, hotel και γίνεται συνένωση αυτών σε ένα dataframe (δομή δεδομένων) το merge_df.
- (5-6) Βρίσκεται η ελάχιστη και η μέγιστη τιμή του Lat (Latitude) από όλες τις εγγραφές του merge_df
- (7-7) Εκτελώντας την παραπάνω πράξη βρίσκεται το εύρος των Lat που έχει κάθε partition
- (8-8) Δημιουργείται μια νέα μεταβλητή (partition_label) στο dataframe merge_df όπου για κάθε εγγραφή έχει μια ετικέτα που δηλώνει σε ποιο partition είναι.
- (9-13) Σε αυτό το κομμάτι του αλγορίθμου υλοποιείται η μεθοδολογία παραγωγής των duplicates που περιεγράφηκε στο κεφάλαιο 3.1. Η μεταβλητή hotel_min αντικατοπτρίζει τα hotels που έχουν μικρότερο Lat από το Lat του σημείου τομής των δεδομένων. Η hotel_max αντικατοπτρίζει τα hotels που έχουν μεγαλύτερο Lat από το σημείο τομής των δεδομένων.
- (14-15) Γίνεται ενοποίηση όλων των dataframe: merge_df, hotel_min, hotel_max σε ένα που ονομάζεται partition_data και εκτελείται η εντολή repartition για τον διαχωρισμό των δεδομένων βάση της μεταβλητής partition_label.
- (16-16) Εκτελείται το cross join μεταξύ hotel και restaurant για τα hotel και restaurant που βρίσκονται στο ίδιο partition.
- Τέλος για κάθε ζευγάρι (restaurant, hotel) που έχει παραχθεί υπολογίζεται η χωρική απόσταση μέσω του αλγορίθμου haversine και η κειμενική ομοιότητα μέσω του αλγορίθμου jaccard.

3.3 Πειραματική Διαδικασία

Η εγκατάσταση του Apache Spark έγινε στο παρακάτω σύστημα. Εκτεταμένη αναφορά για το Apache Spark γίνεται στο Παράρτημα Α

CPU	Intel i7-9750h (6 cores 12 threads)
RAM	16gb DDR4
SDD	512gb m2.sata

Πίνακας 7 Τεχνικά χαρακτηριστικά υπολογιστή

Σε αυτό το σύστημα δημιουργήθηκαν 2 εικονικές μηχανές με την βοήθεια του virtual box μια master και μια slave με τους παρακάτω υπολογιστικούς πόρους

Virtual Machines	CPU	RAM
Master	8 threads	6144 MB
Slave	4 threads	3072 MB

Πίνακας 8 Τεχνικά χαρακτηριστικά Apache Spark Cluster

Το dataset που χρησιμοποιήσαμε είχε τις εξής εγγραφές

Restaurant	Hotel	Restaurant+Hotel
38650	38330	76980

Πίνακας 9 Χωρο-κειμενικά δεδομένα

Για το συγκεκριμένο dataset και τον συγκεκριμένο αλγόριθμο έγινε μια σειρά από δοκιμές στον παραπάνω cluster με διαφορετικές ακτίνες R, διαφορετικό αριθμό partitions και σταθερή τιμή του βαθμού κειμενικής ομοιότητας.

Partitions	R	Records after duplication	Time (with count and python Timer) sec	Time (with count and Spark Timer) sec
2	0.5	76981	4.433	4.007
2	3	76981	4.617	4.007
2	100	115631	4.766	4.008
4	0.5	77068	4.662	4.008
4	3	77150	4.579	4.010
4	100	192931	4.485	4.008
8	0.5	77142	1.874	2.007
8	3	88613	1.976	2.012
8	100	347531	2.095	2.007
12	0.5	85292	1.935	2.021
12	3	112070	1.758	1.010
12	100	502131	1.825	1.011
16	0.5	79047	1.779	1.010
16	3	114952	1.782	1.010
16	100	656731	1.686	1.010

Πίνακας 10 Χρόνος εκτέλεσης στο Apache Spark για κάθε διαφορετικό dataset

Για την διαδικασία της χρονομέτρησης του αλγορίθμου με την εύρεση των δύο αποστάσεων χρησιμοποιήθηκε η εντολή count() που μετρά το πλήθος των εγγραφών με σκοπό να αναγκάσω το spark να προσπελάσει όλες τις εγγραφές του dataframe. Ο χρόνο εκτέλεσης του count() τον πάρθηκε και από τον timer της python και από το Spark Jobs όπου φαίνονται οι εργασίες που έχουν τρέξει στο Spark και η διάρκειά τους. Επίσης δημιουργήθηκε και μια στήλη στον πίνακα με τον αριθμό των εγγραφών μετά την διαδικασία του duplication.

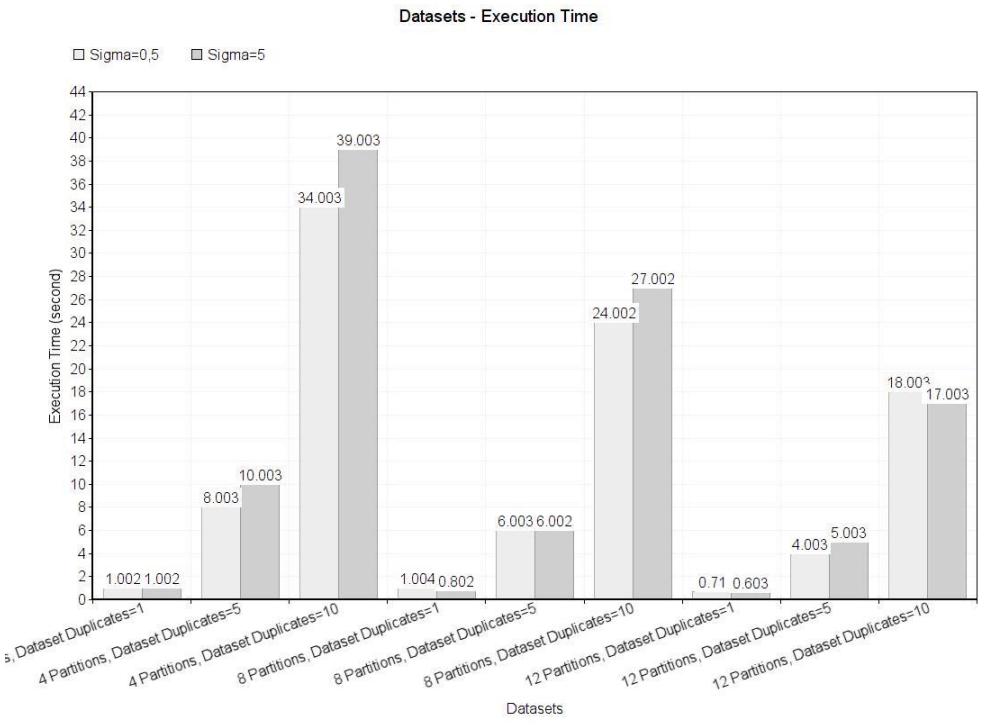
Στο επόμενο στάδιο της πειραματικής διαδικασίας δημιουργήθηκε ένας αλγόριθμος που δέχεται σαν είσοδο 2 dataset για παράδειγμα στην προκειμένη περίπτωση ένα dataset με ξενοδοχεία και ένα με εστιατόρια και δημιουργεί αντίγραφα του αρχικού dataset ανάλογα με το πόσες φορές θέλει ο χρήστης να δημιουργηθούν αντίγραφα των ξενοδοχείων και των εστιατορίων και παράγει ένα ενοποιημένο dataset που αποτελείτε από κ*ξενοδοχεία και λ*εστιατόρια. Επίσης για καθένα από τα 2 dataset κ*ξενοδοχεία, λ*εστιατόρια ο χρήστης έχει την δυνατότητα να ορίσει με την μέση τιμή και τυπική απόκλιση των lon, lat που θέλει να ακολουθούν οι συντεταγμένες των κ*ξενοδοχεία και των λ*εστιατόρια. Επομένως δημιουργήθηκαν τα παρακάτω dataset για επιπλέον δοκιμές του αλγόριθμου που έχει δημιουργηθεί για το spatio-textual join.

Dataset	**Τυπική Απόκλιση (sigma)**	**Αντίγραφα του Αρχικού Dataset για(duplicates)**		**Μέση τιμή των (mean)**	
		Ξενοδοχεία	**Εστιατόρια**	**Lat**	**Lon**
1	0.5	1	1	50	50
2	0.5	5	5	50	50
3	0.5	10	10	50	50
4	5	1	1	50	50
5	5	5	5	50	50
6	5	10	10	50	50

Πίνακας 11 Ιδιότητες dataset

Στη συνέχεια παρουσιάζονται τρία διαγράμματα για τον χρόνο εκτέλεσης των αλγορίθμων, την δημιουργία των duplicates πριν την διαδικασία του partitioning και τον συνολικό αριθμό των εγγραφών που παράγονται.

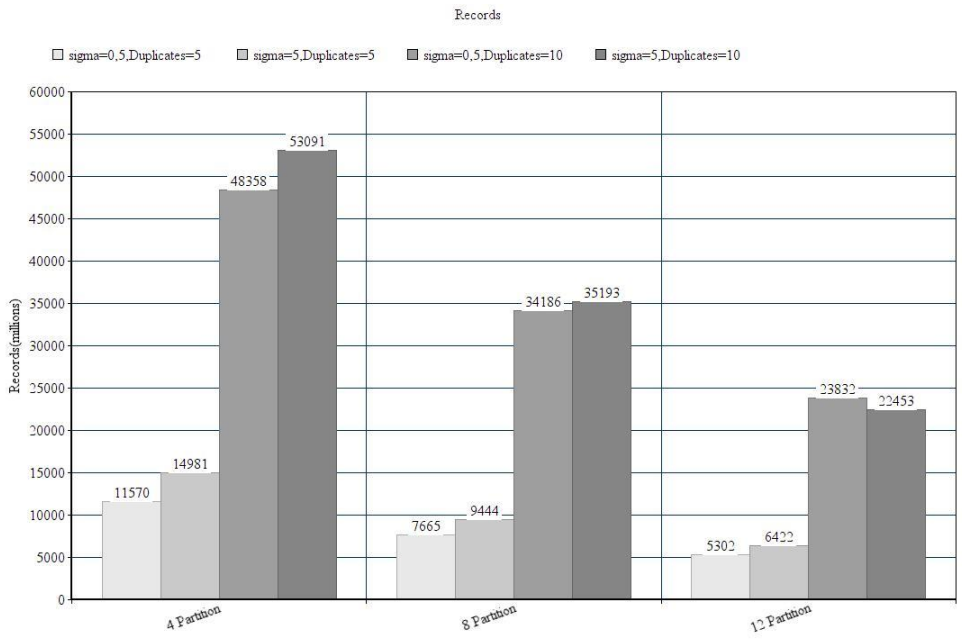
Στο πρώτο διάγραμμα παρατηρείται πως επηρεάζει των χρόνο εκτέλεσης η διαφορετική τιμή του sigma.



Εικόνα 2 Apache Spark Χρόνος εκτέλεσης για κάθε dataset

Συμπεραίνουμε ότι τα δεδομένα όπου τα lat,lon ακολουθούν κανονική κατανομή με sigma=0,5 έχουν ελαφρώς καλύτερους χρόνους εκτέλεσης σε σχέση με τα δεδομένα που ακολουθούν κανονική κατανομή με sigma=5.

Στο επόμενο γράφημα παρουσιάζεται ο συνολικός αριθμός των εγγραφών μαζί με τα duplicates που δημιουργούνται για τις ανάγκες του partitioning.



Εικόνα 3 Apache Spark Πλήθος εγγραφών μετά την εκτέλεση του Cross Join

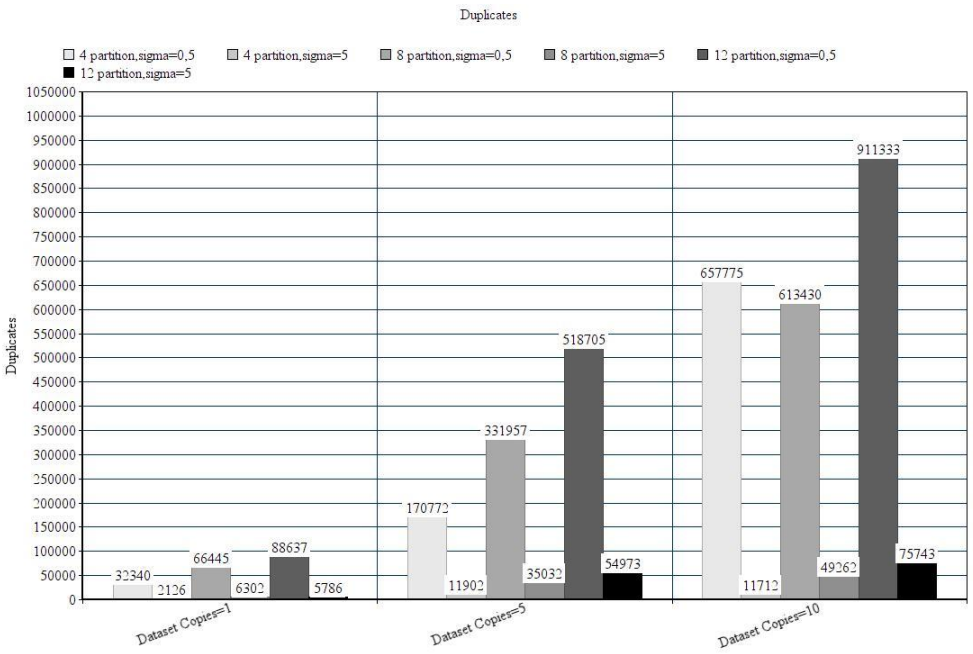
Παρατηρούμε ότι όσο αυξάνεται ο αριθμός των partition οι εγγραφές μειώνονται, το οποίο είναι λογικό διότι οι εγγραφές είναι όλοι οι συνδυασμοί ξενοδοχείων με εστιατορίων ανά partition. Για παράδειγμα έστω ότι έχουμε 20 ξενοδοχεία και 20 εστιατόρια και είναι μοιρασμένα στα partitions με τον εξής τρόπο.

Partitions	**1**	**2**
Ξενοδοχεία	10	10
Εστιατόρια	10	10
Συνδυασμοί	10*10=100	10*10=100

Partitions	**1**	**2**	**3**	**4**
Ξενοδοχεία	5	5	5	5
Εστιατόρια	5	5	5	5
Συνδυασμοί	5*5=25	5*5=25	5*5=25	5*5=25

Αν αθροίσουμε τους συνδυασμούς στα 2 partitions και στα 4 partitions παρατηρούμε ότι στην περίπτωση των 2 έχουμε 200 συνδυασμούς εγγραφές και σε αυτήν των 4 έχουμε 100.

Στο επόμενο γράφημα παρατηρούμε το πλήθος των duplicates που δημιουργούνται για 2 διαφορετικά sigma της κατανομής των lon,lat.



Εικόνα 4 Πλήθος των Duplicates για κάθε Dataset και αριθμό Partition

Παρατηρούμε ότι τα δεδομένα που ακολουθούν κανονική κατανομή με sigma=0,5 έχουν δημιουργήσει πολύ περισσότερα duplicates σε σχέση με τα δεδομένα που ακολουθούν κανονική κατανομή με sigma=5 διότι τα δεδομένα που έχουν sigma=5 είναι καλύτερα διαμοιρασμένα στα partition σε σχέση με αυτά που οι περισσότερες τιμές είναι γύρω από την μέση τιμή.

3.4 Συμπέρασμα

Παρατηρούμε ότι κατά την αύξηση των partition από 4 σε 8 και 12 σύμφωνα με τους χρόνους του Spark υπάρχει μια μείωση των χρόνων στο μισό. Κατά την αύξηση των partition από 12 σε 16 οι χρόνοι εκτέλεσης παραμένουν ίδιοι και αυτό οφείλεται στο γεγονός ότι ο master και ο slave έχουν σύνολο 12 threads οπότε ο βέλτιστος χρόνος εκτέλεσης του προγράμματος προκύπτει όταν ο αριθμός των partitions είναι ο ίδιο με τον αριθμό των threads.

4 Κεφάλαιο - Spatial Join στην κάρτα γραφικών και στο Apache Spark

Χρησιμοποιώντας την λογική του αλγορίθμου που έχει δημιουργηθεί παραμετροποιήθηκε καταλλήλως για να μπορεί να εκτελείται αξιοποιώντας την επεξεργαστική ισχύ της κάρτας γραφικών όπως επίσης και την video ram της κάρτας γραφικών. Για να επιτευχθεί αυτό χρησιμοποιήθηκαν τα εξής: Nvidia Rapids,cuDF,blazingsql,cuspatial.

4.1 Rapids

Το Nvidia Rapids περιλαμβάνει μια συλλογή από βιβλιοθήκες της Nvidia με την χρήση της οποίας έχεις πρόσβαση στην επεξεργαστική ισχύ της κάρτας γραφικών ή σε έναν cluster από κάρτες γραφικών για επεξεργασία δεδομένων σε κατανεμημένο περιβάλλον με κάρτες γραφικών. Για να αξιοποίησει το Nvidia Rapids την κάρτα γραφικών είναι απαραίτητο να χρησιμοποιήσει το Nvidia CUDA για την μετατροπή του υψηλού προγραμματιστικού επιπέδου κώδικα(python) σε χαμηλού.

Το Nvidia Rapids περιλαμβάνει την βιβλιοθήκη cudf η οποία χρησιμοποιήθηκε και είναι η αντίστοιχη pandas της python ώστε διαβάζοντας ένα αρχείο και αποθηκεύοντας αυτό σε ένα dataframe το dataframe να φορτώνεται στην μνήμη της κάρτας γραφικών(video ram) για περαιτέρω επεξεργασία πάνω σε αυτή.

Η βιβλιοθήκη που χρησιμοποιήθηκε για την εκτέλεση queries πάνω σε ένα cudf dataframe είναι η Blazing SQL[2] η οποία σου δίνει την δυνατότητα να συντάσσεις query τύπου sql όμως αυτά σε να εκτελούνται είτε σε ένα κατανεμημένο περιβάλλον από κάρτες γραφικών είτε κεντρικοποιημένα σε μια κάρτα γραφικών.

Σύμφωνα με συγκριτικά πειράματα η blazingsql αποδεικνύεται αρκετές φορές πιο γρήγορη από ένα μοντέλο σε Apache Spark. Ένα τέτοιο πείραμα έχει δημοσιευθεί στο medium από τον Winston Robson.Το πείραμα γίνεται πάνω σε δεδομένα με τη χρήση των οποίων να παραχθεί μοντέλο μηχανικής μάθησης για την εκτίμηση κινδύνου στεγαστικού δανείου. Το πλήθος των δεδομένων είναι 400gb και οι λειτουργίες του πειράματος είναι το λεγόμενο ETL(Extract,Transform,Load). Για την εφαρμογή του πειράματος στην κάρτα γραφικών με χρήση της blazing sql χρησιμοποιήθηκε μια κάρτα γραφικών Nvidia T4. Ενώ για την εφαρμογή του πειράματος σε περιβάλλον Apache Spark χρησιμοποιήθηκαν 4 υπολογιστές αλλιώς 4 κόμβοι.

Οι επιδόσεις τους είναι οι εξής:

Tool	Workload	Time in Seconds
blazingSQL	3.8gb(T4)	32.60
	15.5gb(T4)	132.28
Apache Spark	3.8gb(4 Nodes)	1123.20
	15.6gb(4 Nodes)	2779.70

[2] https://medium.com/future-vision/what-is-blazingsql-33022a677dee

Παρατηρούμε ότι ο χρόνος διεξαγωγής του πειράματος με την χρήση της blazingSQL σε σχέση με το Apache Spark είναι 34 φορές μεγαλύτερος.

Τέλος η βιβλιοθήκη που χρησιμοποιήθηκε και δεν περιλαμβάνεται στο Nvidia Rapids είναι η cuspatial μέσω της οποίας παρέχεται ο αλγόριθμος για την μέτρηση μεταξύ 2 σημείων στον χάρτη μέσω της απόστασης Haversine.

4.2 Spatial Join

Για την εκτέλεση του αλγορίθμου στην κάρτα γραφικών χρειάστηκε τα εξής:

- Διάβασμα του dataset και αποθήκευση αυτού σε ένα cudf dataframe
- Εκτέλεση ενός query για την δημιουργία ενός dataframe που θα αντιστοιχίζεται κάθε ξενοδοχείο με κάθε εστιατόριο(cross join). Δηλαδή κάθε εγγραφή του dataframe περιέχει ένα ξενοδοχείο και ένα εστιατόριο με τα λοιπά χαρακτηριστικά τους.
- Μέτρηση της απόστασης κάθε εγγραφής του νέου dataframe.

Στη συνέχεια παρατίθενται ο ψευδοκώδικας του αλγορίθμου που εκτελέστηκε στην κάρτα γραφικών.

```
1. restaurant = read(file)
2. hotel = read(file)
3. merge_df = hotel.union(restaurant)
4. data = merge_df (restaurant cross_join hotel )
5. for each i in data
6.        haversine_result = haversine(hotel, restaurant)
```

Η επεξήγηση του αλγορίθμου θα γίνει βάση των αριθμού του γραμμών του ψευδοκώδικα.

- (1-3) Γίνεται ανάγνωση των δύο αρχείων και φόρτωση αυτών σε δύο dataframe:restaurant, hotel. Στην συνέχεια τα δύο dataframe ενοποιούνται σε ένα που ονομάζεται merge_df.
- (4-4) Εκτελείται ένα cross join μεταξύ των restaurant και των hotel και δημιουργούνται ζεύγη (restaurant, hotel). Κάθε ζεύγος αποτελεί μια εγγραφή του dataframe data
- (5-6) Για κάθε εγγραφή του dataframe data υπολογίζεται η χωρική απόσταση των ζευγαριών (restaurant, hotel) μέσω του αλγορίθμου haversine

Για την εκτέλεση του αλγορίθμου σε περιβάλλον Apache Spark διατηρήθηκε η ίδια υλοποίηση με αυτή που προαναφέρεται για το spatio-textual join χωρίς να υπολογίζει την κειμενική ομοιότητα μεταξύ 2 αντικειμένων και κατ' επέκταση και τα δεδομένα δεν έχουν καθόλου πληροφορία κειμένου. Η λογική του partitioning που είχε προαναφερθεί διατηρήθηκε η ίδια και σε αυτόν τον αλγόριθμο.

Η παραπάνω διαφορές και ομοιότητες παρατηρούνται και στον παρακάτω ψευδοκώδικα.

```
1.  partiotion_number=8
2.  restaurant=read(file)
3.  hotel=read(file)
4.  merge_df=hotel.union(restaurant)
5.  max_lat=merge_df("Lat").max()
6.  min_lat=merge_df("Lat").min()
7.  partition=(max_lat-min_lat)/partition_number
8.  merge_df=merge_df.withcolumn(partition_label for each record)
9.  for each I in partition_number:
10.    hotel_min=merge_df.select(*).where(lat>max_lat – (partition_number – i) * partition) – R)
11.    AND lat< maxl_lat-(partition_number – I ) * partition)))
12.    hotel_max=merge_df.select(*).where(lat<max_lat – (partition_number – i) * partition) + R)
13.    AND lat> maxl_lat-(partition_number – i) * partition)))
14. partition_data = merge_df.union(hotel_min).union(hotel_max)
15. partition_data = partition_data.repartition(partition_label)
16. partition_data = partition_data(restaurant cross_join hotel by partition)
17. for each i in partition_data
18.       haversine_result = haversine(hotel, restaurant)
```

Σε σχέση με τον ψευδοκώδικα που είχε επισυναφθεί στο κεφάλαιο 3.2 δεν υπάρχει μόνο η γραμμή 19 όπου εκτελούνταν ο αλγόριθμος jaccard που υπολογίζει την κειμενική ομοιότητα.

4.3 Πειραματική διαδικασία

4.3.1 Dataset

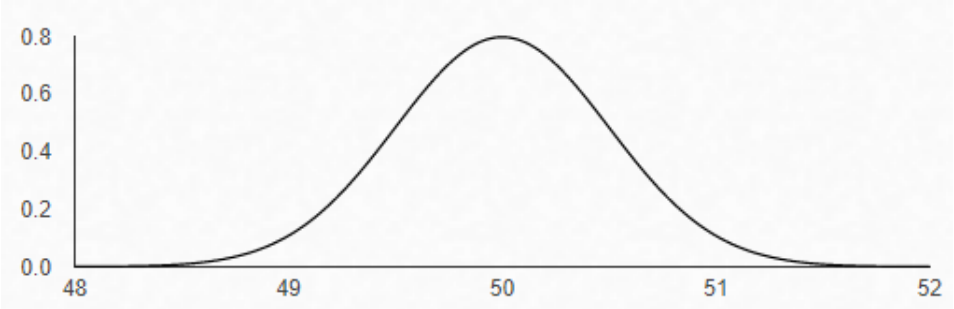
Για την διεξαγωγή την πειραματικής διαδικασίας δημιουργήθηκαν datasets που εκτελέστηκαν και σε περιβάλλον Apache Spark και σε περιβάλλον Nvidia.Τα dataset περιέχουν συντεταγμένες 2 διαφορετικών αντικειμένων. Στην συγκεκριμένη περίπτωση περιέχουν ξενοδοχεία και εστιατόρια και το ερώτημα είναι να βρεθεί η απόσταση haversine όλων των ξενοδοχείων από όλων των εστιατορίων. Όλα τα dataset ακολουθούν το ίδιο σχήμα και αυτό που διαφέρει μεταξύ τους είναι το πλήθος των εγγραφών τους και η κατανομή των συντεταγμένων τους. Το σχήμα των datasets είναι το εξής:

ID	Name	Label	Lon	Lat

Επεξήγηση των μεταβλητών του σχήματος:

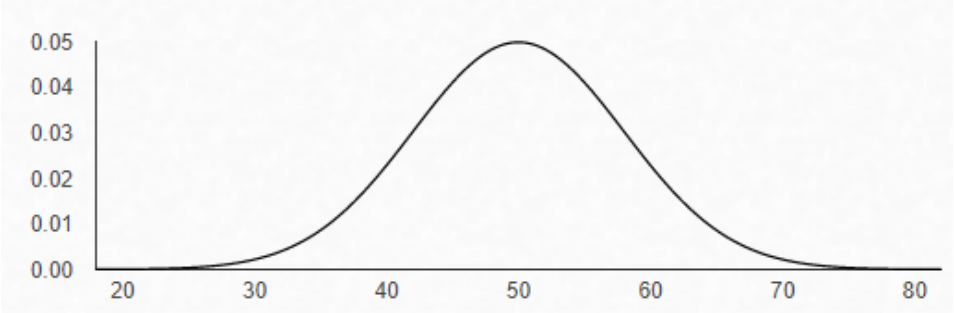
- ID: Συμβολίζει τον μοναδικό αριθμό id κάθε εγγραφής.
- Name: Είναι το όνομα κάθε εστιατορίου ή ξενοδοχείου
- Label: Είναι η ετικέτα κάθε εγγραφής όπου με αυτή δηλώνεται αν είναι ξενοδοχείου ή εστιατόριο. Οι τιμές της είναι H για ξενοδοχεία και R για εστιατόρια
- Lon: Είναι συντομογραφία της λέξης longitube και απεικονίζει το γεωγραφικό μήκος κάθε εστιατορίου ή ξενοδοχείου
- Lat: Είναι συντομογραφία της λέξης latitube και απεικονίζει το γεωγραφικό πλάτος κάθε εστιατορίου ή ξενοδοχείου

Τα dataset που δημιουργήθηκαν όπως προαναφέρθηκε έχουν διαφορετικό πλήθος εγγραφών συνολικά εστιατορίων και ξενοδοχείων με αναλογία 50% εστιατόρια και 50% ξενοδοχεία. Τα διαφορετικά πλήθη εγγραφών είναι τα εξής:2530,3577,4381,5060,5639,6200. Επίσης για κάθε ένα από τα dataset δημιουργήθηκαν δύο είδη κατανομών των τιμών των lat,lon. Στην μία περίπτωση το σ που είναι η τυπική απόκλιση είναι σ=0,5 και στην άλλη σ=8 με μέση τιμή μ=50. Οπότε η κατανομή των τιμών του lon,lat για σ=0,5 είναι:



Εικόνα 5 Κανονική κατανομή για sigma=0,5

Για σ=8 είναι:



Εικόνα 6 Κανονική κατανομή για sigma=8

Συνοψίζοντας η ποικιλία των dataset αποτυπώνεται στον παρακάτω πίνακα.

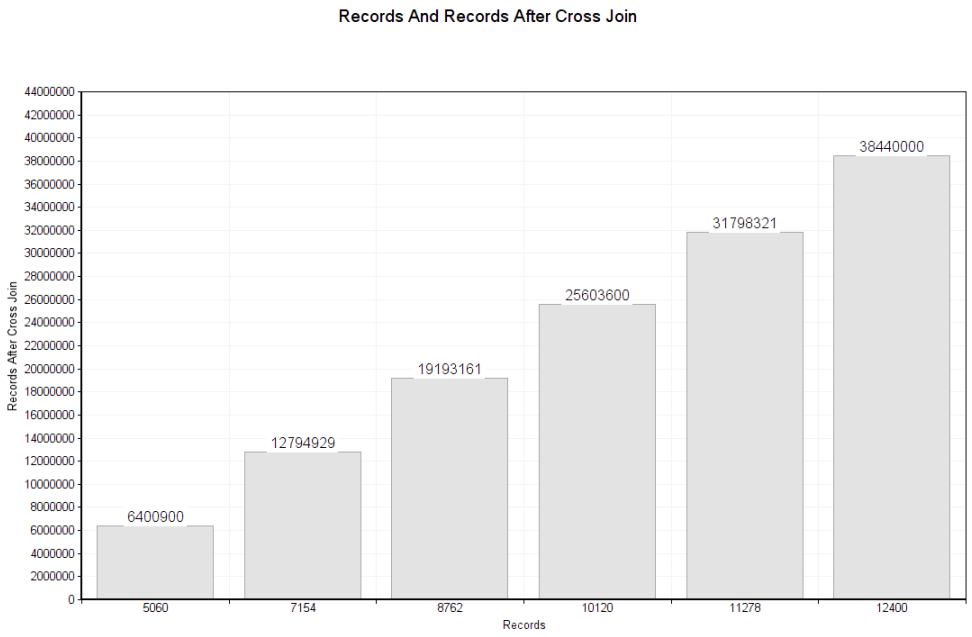
Dataset	Πλήθος Εγγραφών	Τυπική Απόκλιση (sigma)	Πλήθος Εστιατορίων, Ξενοδοχείων		Μέση τιμή (mean)	
			Ξενοδοχεία	Εστιατόρια	Lat	Lon
1	5060	0.5	2530	2530	50	50
2	5060	8	2530	2530	50	50
3	7154	0.5	3577	3577	50	50
4	7154	8	3577	3577	50	50
5	8762	0.5	4381	4381	50	50
6	8762	8	4381	4381	50	50
7	10120	0.5	5060	5060	50	50
8	10120	8	5060	5060	50	50
9	11278	0.5	5639	5639	50	50
10	11278	8	5639	5639	50	50
11	12400	0.5	6200	6200	50	50
12	12400	8	6200	6200	50	50

4.3.2 Εκτέλεση της πειραματικής διαδικασίας

Για την διεξαγωγή της πειραματικής διαδικασίας στην κάρτα γραφικών χρησιμοποιήθηκε μια Nvidia RTX2060 με τα εξής χαρακτηριστικά:

Cuda cores	1920
Memory Speed	14Gbps
Memory Bandwidth	336 GB/s
Video Ram	6bGB GDDR6

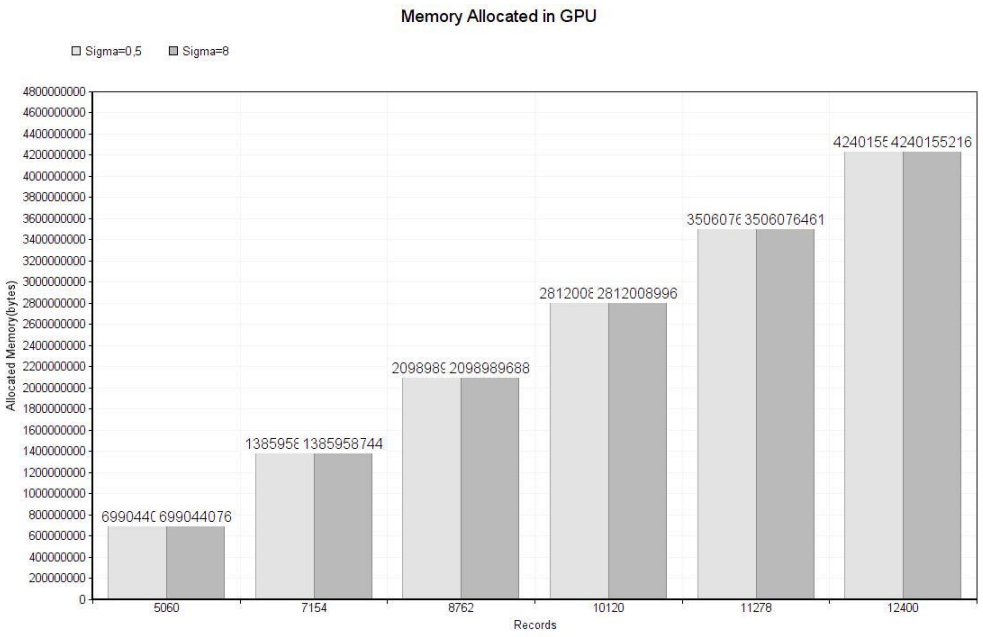
Ξεκινώντας την πειραματική διαδικασία χωρίς να γίνεται κάποιου είδους partitioning επισυνάπτεται ένα γράφημα όπου προβάλλεται το πλήθος των εγγραφών του αρχικού dataset και το πλήθος των εγγραφών όπως προκύπτει μετά την διαδικασία του cross join.



Εικόνα 7 GPU Εγγραφές μετά την εκτέλεση του Cross Join

Το πλήθος των εγγραφών που προκύπτει μετά το cross join είναι από το σύνολο των εγγραφών που είναι αθροιστικά εστιατόρια και ξενοδοχεία σε βαθμό 50% εστιατόρια και 50% ξενοδοχεία άρα: 5060/2=2530. Επομένως έχουμε 2530 εστιατόρια και 2530 ξενοδοχεία επομένως η σύζευξη όλων των ξενοδοχείων με όλων των εστιατορίων είναι:2530*2530=6400900 εγγραφές.

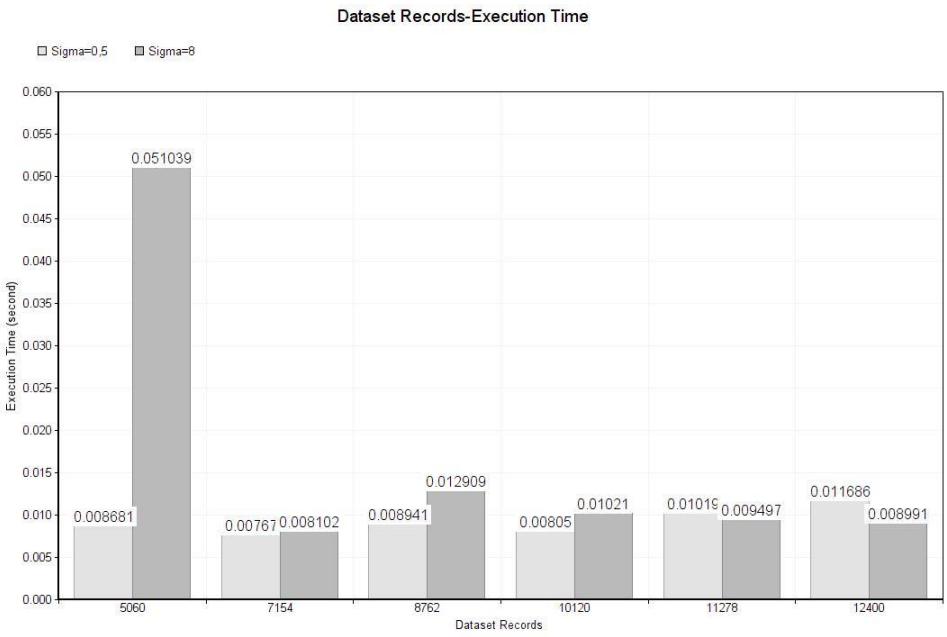
Στη συνέχεια επισυνάπτεται γράφημα που προβάλλει το κάθε dataset πόσο χώρο δεσμεύει στην κάρτα γραφικών.



Εικόνα 8 GPU Δέσμευση μνήμης

Όπως έχει αναφερθεί για κάθε πλήθος εγγραφών των datasets δημιουργούμε δύο datasets ένα με τιμές lat, lon που ακολουθούν την κανονική κατανομή με sigma=0,5 και με sigma=8. Παρατηρούμε πως για dataset με το ίδιο πλήθος εγγραφών αλλά με διαφορετικό sigma ο χώρος που δεσμεύουν στην μνήμη μετά το cross join είναι ακριβώς ο ίδιος και αυτό είναι λογικό γιατί δεν αλλάζει το πλήθος των εγγραφών αλλά οι τιμές του lat, lon.

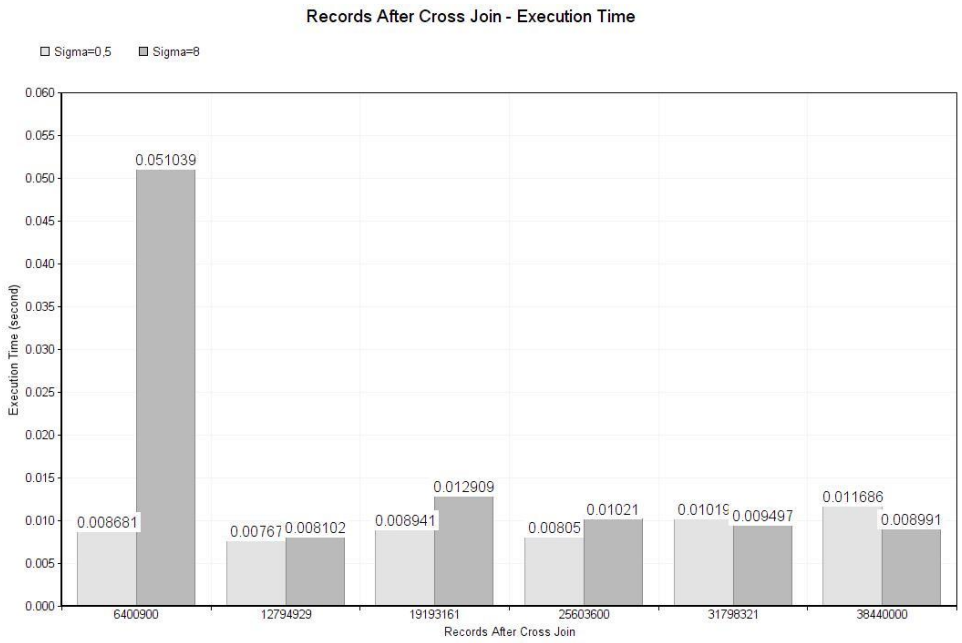
Για το πλήθος των dataset μετρήθηκε ο χρόνος εκτέλεσής τους. Ο χρόνος εκτέλεσης μετρήθηκε με τον ίδιο τρόπο που είχε μετρηθεί και στο Apache Spark δηλαδή μετρώντας πόσο χρόνο θα κάνει να τρέξει η εντολή count(), η οποία μετρά το πλήθος των εγγραφών ενός dataframe. Ο σκοπός της χρήσης της count() ήταν διότι η count() είναι μια από τις εντολές που για να εξάγει και εμφανιστεί το αποτέλεσμα πρέπει να προσπελάσει όλες τις εγγραφές.



Εικόνα 9 GPU Αριθμός εγγραφών Dataset - Χρόνο εκτέλεσης

Παρατηρούμε από το παραπάνω γράφημα ότι στα μικρότερα dataset μέχρι το πλήθος των 10200 εγγραφών, απαιτούν λίγο περισσότερο χρόνο εκτέλεσης. Τα dataset όπου οι μεταβλητές lat, lon ακολουθούν κανονική κατανομή με sigma=8 δηλαδή οι τιμές των μεταβλητών δεν είναι πυκνά κατανεμημένες όσο είναι για sigma=0,5.

Στη συνέχεια επισυνάπτεται γράφημα μεταξύ των εγγραφών που προκύπτουν μετά την διαδικασία του cross join σε συσχέτιση με τον χρόνο εκτέλεσης.



Εικόνα 10 GPU Εγγραφές μετά την εκτέλεση του Cross Join – Χρόνος εκτέλεσης

Από το παραπάνω γράφημα παρατηρούμε ότι παρότι το μέγεθος του dataframe μεγαλώνει και καταλαμβάνει περισσότερο χώρο στην μνήμη ο χρόνος εκτέλεσης δεν μεταβάλλεται αρκετά και επίσης δεν υπάρχει κάποια λογική αύξηση αυτού ανάλογη με το μέγεθος του dataframe.

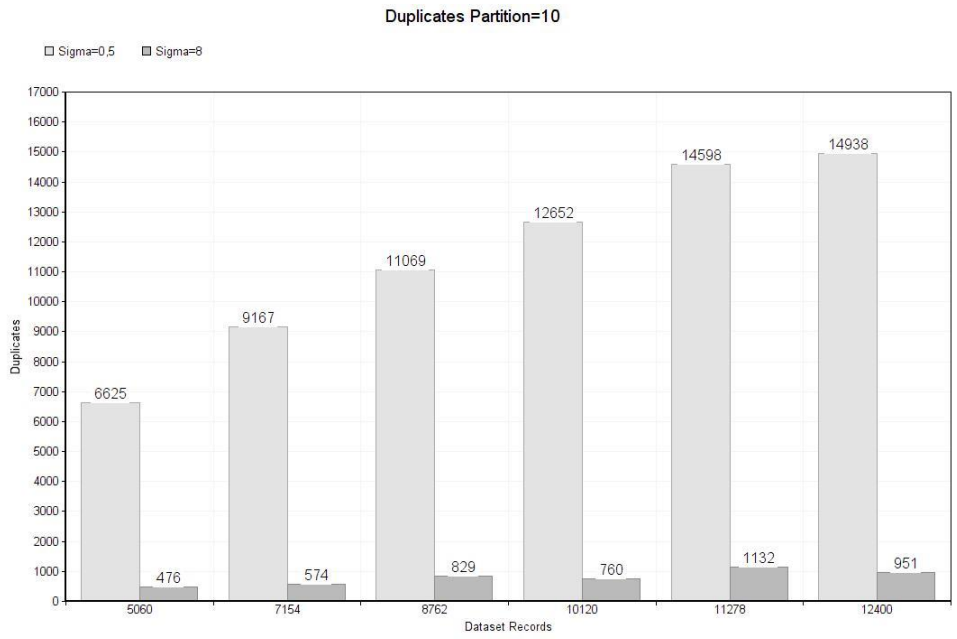
4.3.3 Apache Spark

Τα ίδια dataset που χρησιμοποιήθηκαν για την εκτέλεση του αλγορίθμου spatial join στην κάρτα γραφικών χρησιμοποιήθηκαν και σε περιβάλλον Apache Spark. Με την διαφορά ότι στο Apache Spark χρησιμοποιήθηκε η τεχνική partitioning στην εκτέλεση του αλγορίθμου. Η τεχνική του partitioning που εφαρμόστηκε είναι ακριβώς η ίδια με την τεχνική που είχε εφαρμοστεί στον αλγόριθμο για το spatio-textual join. Δηλαδή ο διαχωρισμός των δεδομένων σύμφωνα με την μεταβλητή lat που επιτυγχάνει να χωρίζει σε κάθετες λωρίδες.

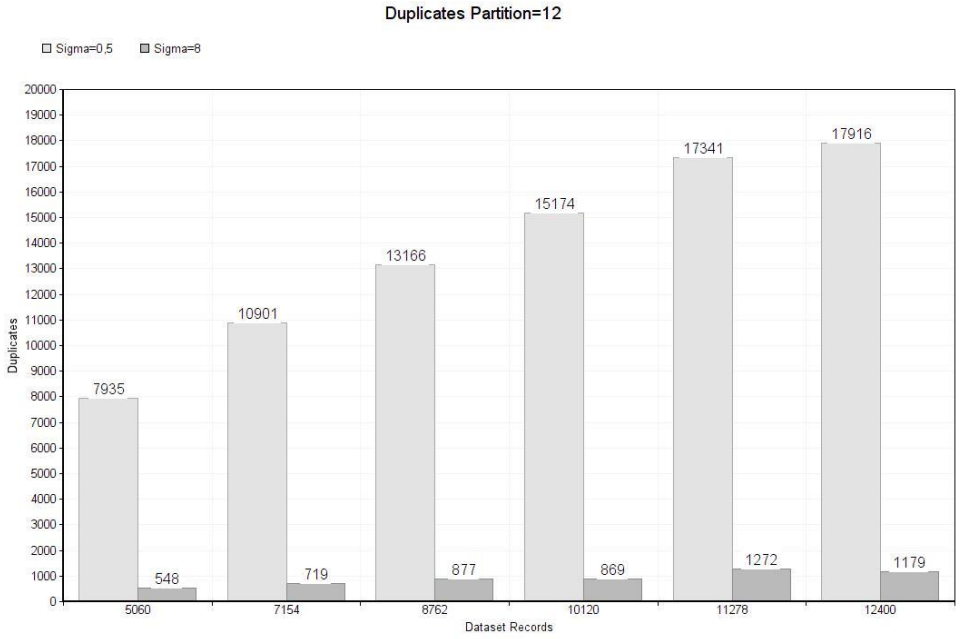
4.3.4 Duplicates

Όπως έχει αναφερθεί χωρίζοντας τα δεδομένα σε partition παρατηρήθηκε ότι στο ένα από τα δύο αντικείμενα του dataset είτε στα ξενοδοχεία είτε στα εστιατόρια υπήρχε η ανάγκη πριν το partitioning να γίνει δημιουργία αντιγράφων(duplicates) ώστε να μην υπάρχει απώλεια πληροφορίας από πιθανά ζευγάρια εστιατορίων-ξενοδοχείων που βρίσκονται εντός μια απόστασης R που έχουμε ορίσει. Στην συγκεκριμένη περίπτωση το R=0,5.

Στη συνέχεια θα προβληθεί η συμπεριφορά των duplicates ανά ίδιο αριθμό partition στα διαφορετικά dataset που έχουν δημιουργηθεί. Στην συγκεκριμένη περίπτωση δημιουργούνται αντίγραφα από ξενοδοχεία.



Εικόνα 11 Apache Spark Duplicates Partition=10

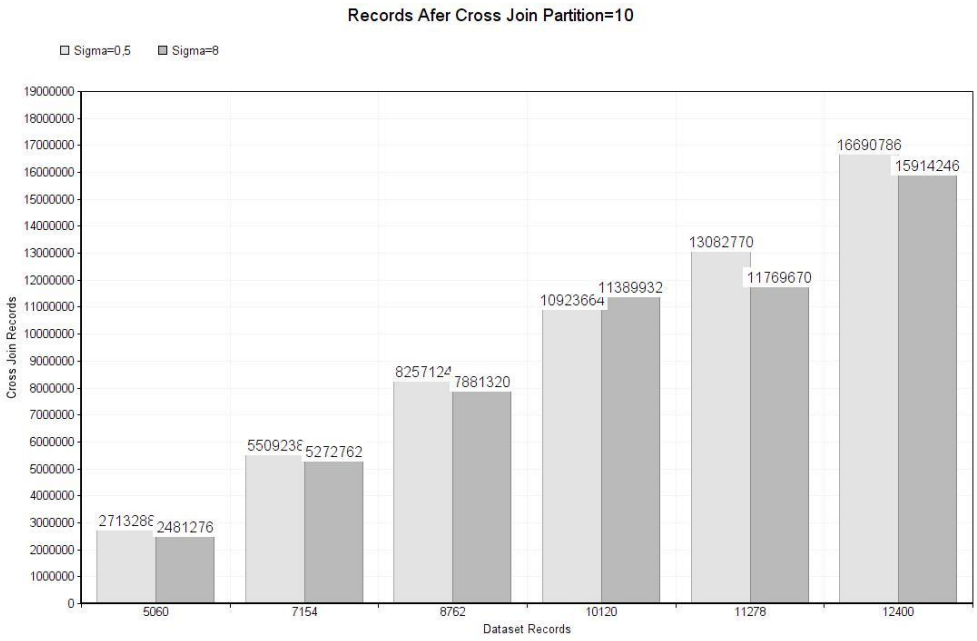


Εικόνα 12 Apache Spark Duplicates Partition=12

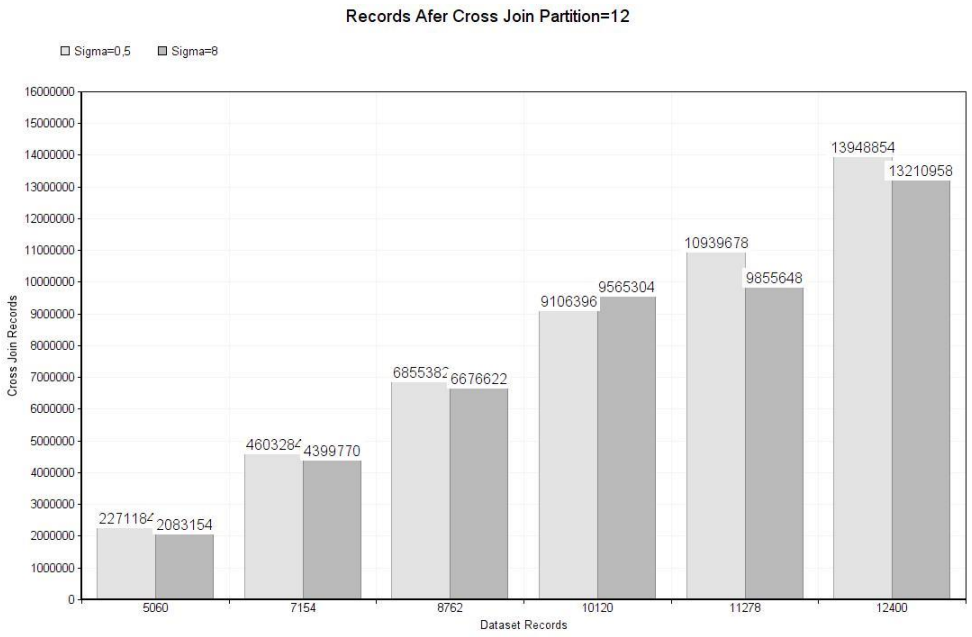
Από τα παραπάνω διαγράμματα παρατηρούμε ότι για οποιοδήποτε αριθμό partition επιλέξαμε και για οποιοδήποτε dataset ο αριθμός των duplicates έχει μεγάλες διαφορές για τα dataset που η μεταβλητή

lat ακολουθεί κανονική κατανομή με sigma=0,5 σε σχέση με αυτά που η μεταβλητή lat ακολουθεί κανονική κατανομή με sigma=8. Αυτό προκύπτει από το γεγονός ότι στην περίπτωση που η κατανομή του lat έχει sigma=0,5 οι τιμές γύρω από την μέση τιμή είναι πολύ πιο πυκνές σε σχέση με την κατανομή με sigma=8. Αυτό παρατηρείται εύκολα και από τα διαγράμματα που επισυνάφθηκαν για τις κατανομές των dataset με δύο διαφορετικά sigma.

Στη συνέχεια εξετάστηκε το πλήθος των εγγραφών που προκύπτουν αφού εκτελεστεί το cross join των εστιατορίων με των ξενοδοχείων σε τέσσερα διαφορετικά ανάλογα με το πλήθος των partition που έχουν χωριστεί κάθε φορά τα δεδομένα. Συγκεκριμένα σε partition=10,12.



Εικόνα 13 Apache Spark Εγγραφές μετά την εκτέλεση του Cross Join Partition=10



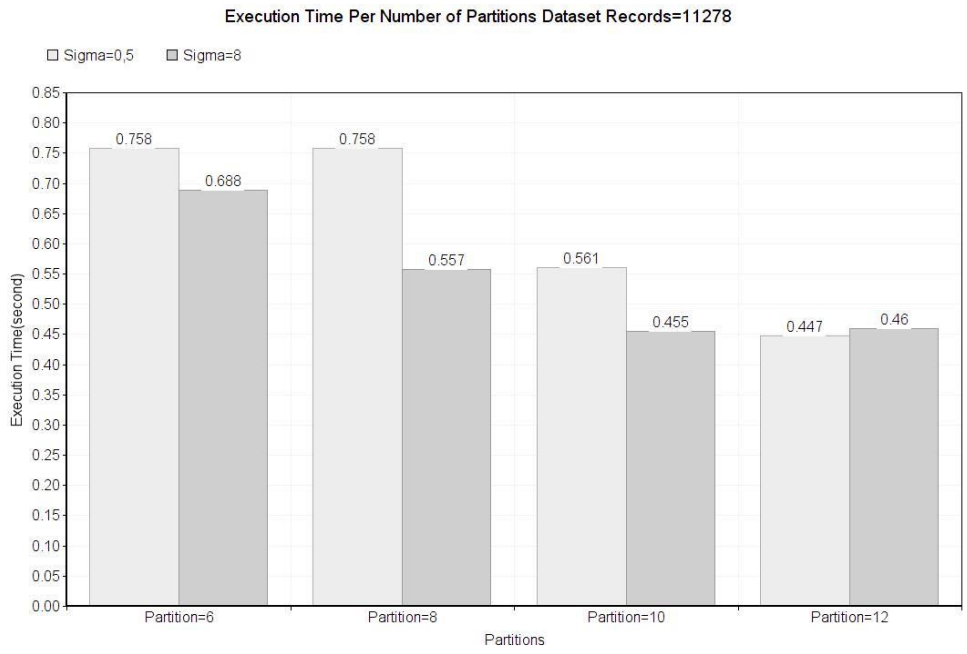
Εικόνα 14 Apache Spark Εγγραφές μετά την εκτέλεση του Cross Join Partition=12

Παρατηρούμε ότι παρότι τα dataset με διαφορετικό sigma έχουν μεγάλη διαφορά στον αριθμό των duplicates και θα αναμέναμε εξίσου μεγάλη διαφορά και στον αριθμό των εγγραφών που προκύπτουν μετά την εκτέλεση του cross join παρατηρούμε ότι αυτό δεν υφίσταται. Βέβαια σχεδόν σε όλες τις περιπτώσεις παραμένει μεγαλύτερο το πλήθος των εγγραφών για τα dataset που ακολουθεί το lat τους κατανομή με sigma=0,5 σε σχέση με τα dataset που ακολουθεί κατανομή με sigma=8. Επίσης παρατηρούμε ότι καθώς αυξάνεται το πλήθος εγγραφών του dataset αυξάνεται και το πλήθος των εγγραφών μετά την εκτέλεση του cross join. Είναι αναμενόμενο το συγκεκριμένο αποτέλεσμα διότι αφού αυξάνουμε το πλήθος των ξενοδοχείων και των εστιατορίων είναι λογικό και το γινόμενο αυτό των 2 να αυξηθεί.

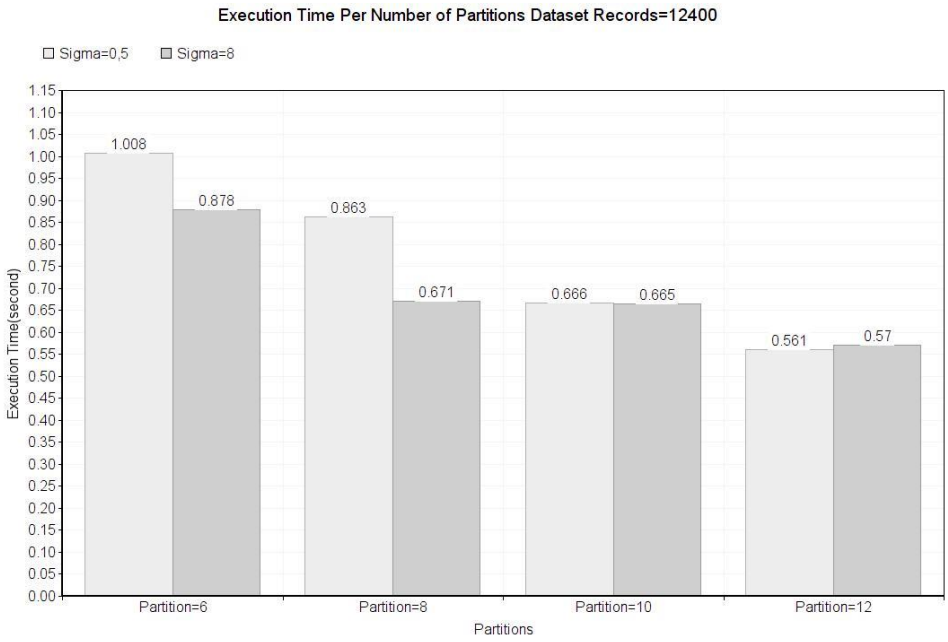
Στη συνέχεια εξετάστηκε ο χρόνος εκτέλεσης που απαιτεί κάθε dataset σε περιβάλλον Apache Spark. Συγκρίθηκαν τα dataset που με ίδιο αριθμό εγγραφών και διαφορετικό sigma. Επίσης συγκρίνεται πώς διαμορφώνεται ο χρόνος εκτέλεσης με την αύξηση του πλήθους των partition.

Όπως έχει αναφερθεί ο χρόνος εκτέλεσης μετρήθηκε μέσω της εντολής count λόγω του ότι για να παράγει αποτέλεσμα πρέπει να προσπελάσει όλες τις εγγραφές και μας διασφαλίζει ότι επεξεργασία έχει προηγηθεί πριν στα data έχει εκτελεστεί. Ο χρόνος εκτέλεσης της count πάρθηκε από το Apache Spark που δείχνει κάθε εντολή που έχει εκτελεστεί πόσο χρόνο απαιτεί.

Επισυνάπτονται τα διαγράμματα για διαφορετικό πλήθος εγγραφών του dataset. Σε κάθε γράφημα αναγράφεται στον τίτλο του το πλήθος εγγραφών του dataset στο οποίο έγιναν οι δοκιμές.



Εικόνα 15 Χρόνος εκτέλεσης για κάθε αριθμό partitions Εγγραφές Dataset=11278



Εικόνα 16 Χρόνος εκτέλεσης για κάθε αριθμό partitions Εγγραφές Dataset=12400

Παρατηρούμε σε όλα ότι με την αύξηση των partition ο χρόνος μειώνεται. Αναμενόμενο διότι η εργασία επιμερίζεται σε περισσότερους κόμβους. Βέβαια οι διακυμάνσεις του χρόνου εκτέλεσης είναι πολύ

μικρές και αυτό οφείλεται στο μέγεθος των dataset. Μια καλύτερη εικόνα παρατηρείται στο μεγαλύτερο dataset με τις 12400 εγγραφές όπου απεικονίζονται καλύτερα οι μεταβολές του χρόνου κατά την αύξηση των partition.

4.3.5 Nvidia vs Apache Spark

Στην συνέχεια θα προχωρήσουμε στην σύγκριση των χρόνων εκτέλεσης του αλγορίθμου για το spatial join στην κάρτα γραφικών και στο Apache Spark.Για την εγκατάσταση του Apache Spark χρησιμοποιήθηκε ο υπολογιστής με τα παρακάτω χαρακτηριστικά.

CPU	Intel i7-9750h (6 cores 12 threads)
RAM	16gb DDR4
SDD	512gb m2.sata

Στο υπολογιστή αυτό δημιουργήθηκαν 2 εικονικές μηχανές μια Master και μια Slave και δόθηκαν οι παρακάτω πόροι ώστε να μπορέσει να λειτουργήσει το Apache Spark σε περιβάλλον cluster.

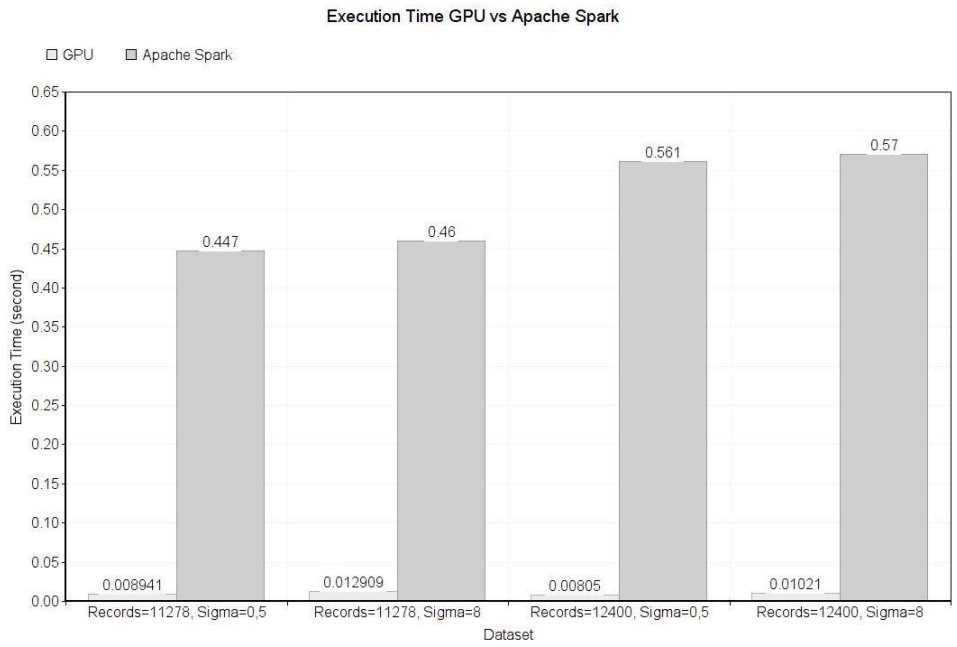
Virtual Machines	CPU	RAM
Master	8 threads	6144 MB
Slave	4 threads	3072 MB

Για την υλοποίηση σε κάρτα γραφικών χρησιμοποιήθηκε μια Nvidia RTX 2060 όπου έχει τα παρακάτω χαρακτηριστικά

Cuda cores	1920
Memory Speed	14Gbps
Memory Bandwidth	336 GB/s
Video Ram	6bGB GDDR6

Πίνακας 12 GPU Specification

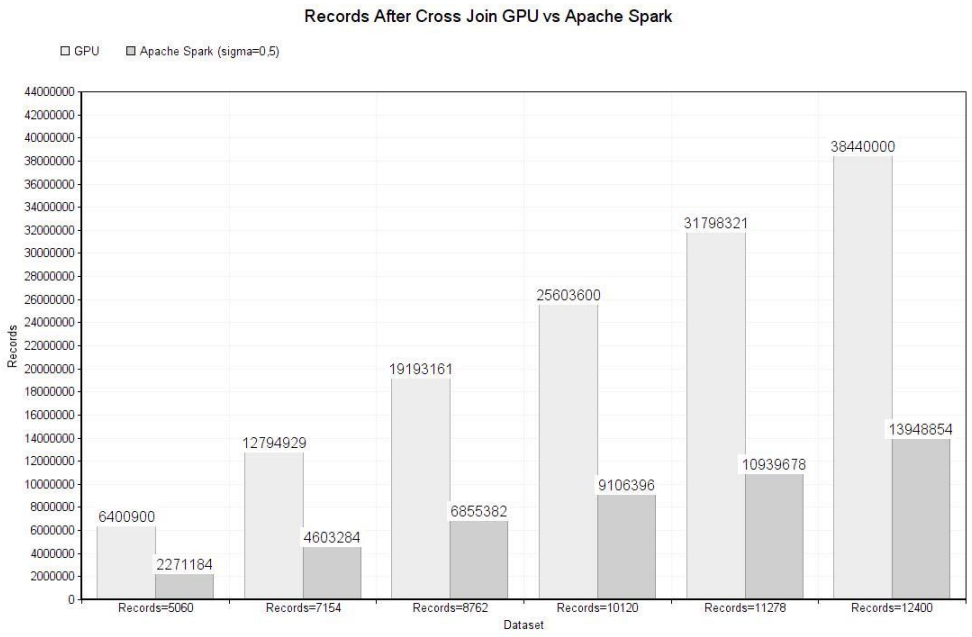
Για την σύγκριση του χρόνου εκτέλεσης στο περιβάλλον Apache Spark με την κάρτα γραφικών θα ληφθεί υπόψη ο καλύτερος χρόνος εκτέλεσης του αλγορίθμου ανά dataset και ανά sigma. Ο καλύτερος χρόνος είναι αυτός όπου τα δεδομένα διαμοιράστηκαν σε 12 partition.



Εικόνα 17 Χρόνος εκτέλεσης GPU vs Apache Spark

Παρατηρείται ότι ο χρόνος εκτέλεσης στην κάρτα γραφικών είναι αρκετά πιο μικρός σε σχέση με το Apache Spark που εκμεταλλεύεται τον επεξεργαστή, την ram και σε μερικές περιπτώσεις αν γεμίσει η ram τον σκληρό δίσκο. Βέβαια το αποτέλεσμα είναι αναμενόμενο βασιζόμενη στην έρευνα που έχει επισυναφθεί με τα δεδομένα που περιέχουν πληροφορίες δανείων. Να σημειωθεί επίσης ότι τα δεδομένα στην κάρτα γραφικών δεν υποβλήθηκαν σε κανενός είδους partitioning. Στην ουσία ο αλγόριθμος εκτελέστηκε κεντρικοποιημένα και ούτε υπήρχε κάποιου είδους file system όπως στο Apache Spark που υπήρχε το HDFS file system. Λόγω των αυξημένων επιδόσεων στης κάρτας γραφικών σε σχέση με το Apache Spark που χρησιμοποιεί τον επεξεργαστή. Η εταιρεία παραγωγής του Apache Spark στην τελευταία έκδοση του περιβάλλοντος 3.0.0 και μετά το Apache Spark να μπορεί να εκτελεστεί σε cluster από κάρτες γραφικών.

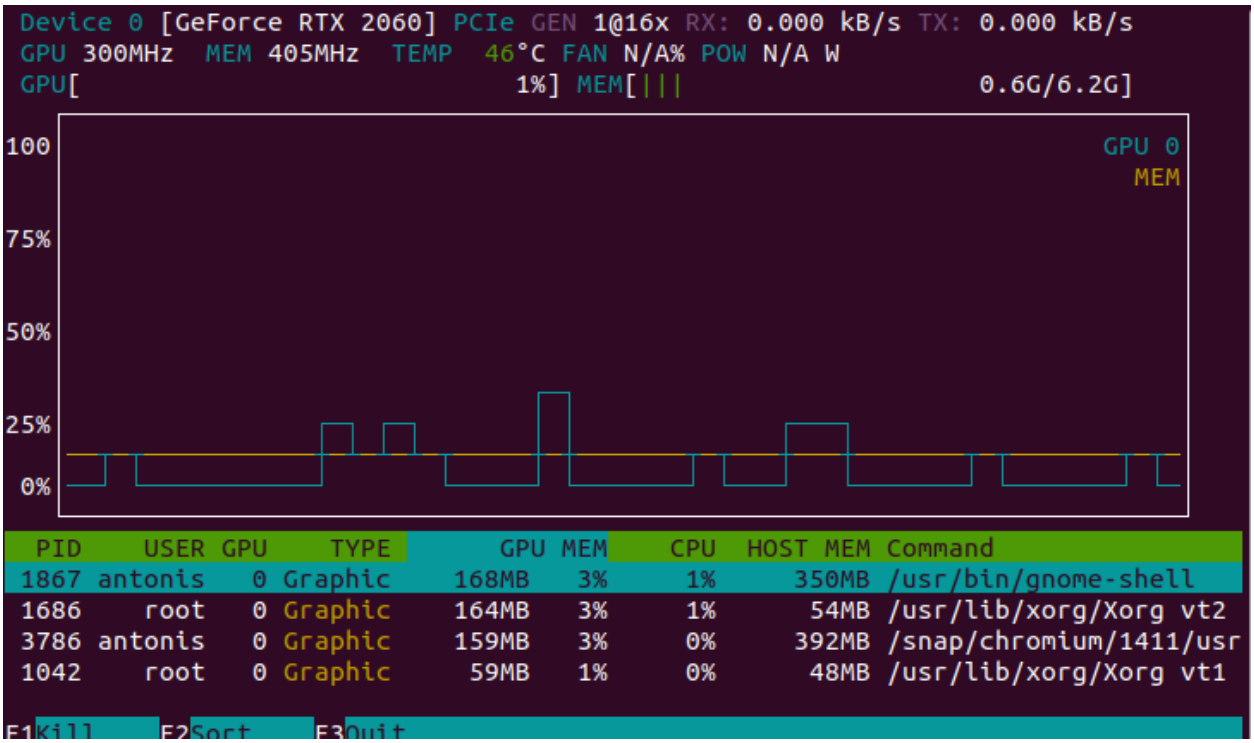
Μετά την σύγκριση των χρόνων εκτέλεσης αξιοσημείωτο να συγκριθεί είναι το πλήθος των εγγραφών που προκύπτει μετά την εκτέλεση της εντολής cross join. Για την σύγκριση επιλέχθηκαν τα dataset που ακολουθεί κανονική κατανομή η μεταβλητή lat με sigma=0,5. Διότι σε περιβάλλον Apache Spark τα συγκεκριμένα dataset έχουν μεγαλύτερο αριθμό από duplicates και κατ' επέκταση προκύπτουν περισσότερες εγγραφές μετά την εκτέλεση της εντολής cross join. Όσο αφορά την επιλογή των dataset για την εκτέλεση του αλγορίθμου στην κάρτα γραφικών δεν επηρεάζει το sigma το πλήθος των τιμών που προκύπτουν από την εκτέλεση του cross join διότι δεν γίνεται κάποιο είδους partitioning οπότε δεν υπάρχουν duplicates.



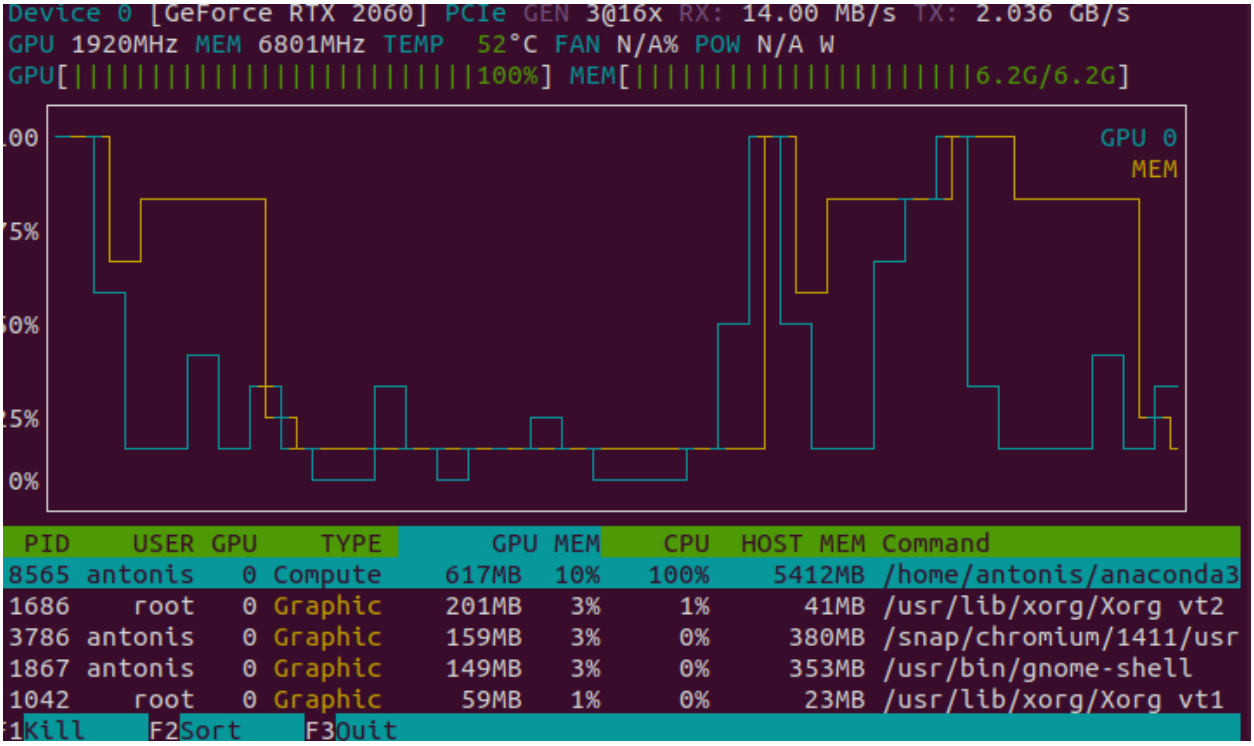
Εικόνα 18 Εγγραφές μετά την εκτέλεση του Cross Join GPU vs Apache Spark

Επίσης ο όγκος των δεδομένων που κλήθηκε να διαχειριστεί η κάρτα γραφικών σε σχέση με το Apache Spark είναι αντιστρόφως ανάλογη με τον χρόνο εκτέλεσης. Στην κάρτα γραφικών επειδή δεν υπήρχε κάποια τεχνική partitioning και στα δεδομένα και όλα τα ξενοδοχεία έκαναν cross join με όλα τα εστιατόρια ο όγκος του dataframe που προέκυπτε είναι αρκετά μεγαλύτερος.

Αξιοσημείωτη είναι επίσης η παρατήρηση του χώρου της video ram που δεσμεύει το cudf dataframe. Για αυτό επισυνάπτονται δύο screenshot από monitoring πρόγραμμα της κάρτας γραφικών και συγκεκριμένα από το nvtop. Στην πρώτη εμφανίζονται οι πόροι που χρησιμοποιούνται πριν την εκτέλεση του αλγορίθμου και στην δεύτερη εμφανίζονται οι πόροι κατά την εκτέλεση του αλγορίθμου.



Εικόνα 19 Πόροι που χρησιμοποιεί η GPU πριν την εκτέλεση του αλγορίθμου



Εικόνα 20 Πόροι που χρησιμοποιεί η GPU κατά την εκτέλεση του αλγορίθμου

Παρατηρούμε ότι πριν την εκτέλεση του αλγορίθμου ο χώρος της video ram που δεσμεύεται είναι 0,6GB/6,2GB σύμφωνα με την ένδειξη επάνω αριστερά που λέγεται MEM και κατά την εκτέλεση του αλγορίθμου η ένδειξη είναι 6,2GB/6.2GB. Επομένως ο αλγόριθμος χρησιμοποιεί 6,2-0,6=5,6 GB. Επίσης για την μέτρηση του ίδιου dataframe χρησιμοποιήθηκε η python εντολή memory_usage(deep=True) και η ένδειξη της είναι η εξής:4240155216 bytes που σημαίνει 4,2 GB. Δηλαδή η απόκλιση της μέτρησης του προγράμματος nvtop σε σχέση με την εντολή memory_usage είναι 5,6-4,2=1,4GB. Το γεγονός αυτό προκύπτει από το ότι η γλώσσα προγραμματισμού που είναι γραμμένο το CUDA στο οποίο εκτελούνται η Blazingsql ,η cuspatial και η cudf είναι η C++ οπότε προγραμματίζοντας σε python. Η python θα πρέπει να μεταγλωττιστεί σε C++ και στη συνέχεια να εκτελεστεί. Μέσω αυτή της διαδικασίας δεσμεύονται επιπλέον πόροι είτε στον επεξεργαστή της κάρτας γραφικών είτε στην ram της που κοστίζουν σε επιδόσεις. Επομένως αν χρειαστεί η εκμετάλλευση της κάρτας γραφικών στον μέγιστο βαθμό η προτεινόμενη γλώσσα προγραμματισμού είναι η C++ με κόστος απέναντι στον χρήστη τον χρόνο εκμάθησης που απαιτεί σε σχέση με την python.

4.4 Συμπεράσματα

Κατά την διαδικασία της εκτέλεσης πειραμάτων σε περιβάλλον Apache Spark παρατηρήθηκε ότι αν δηλωθούν περισσότερα partitions από τα threads που διαθέτει ο επεξεργαστής ή οι επεξεργαστές, ο χρόνος παύει να μειώνεται οπότε μας οδηγεί στο συμπέρασμα ότι το Apache Spark κάνει κάποιου είδους threading. Σε αντίθεση με την κάρτα γραφικών όπου παρότι έγιναν προσπάθειες να γίνει partitioning στα δεδομένα επειδή δεν υπήρχε διαθέσιμη δεύτερη κάρτα δεν υποστηριζόταν το partitioning, άρα δεν υπάρχει η δυνατότητα να γίνει κάποιου είδους threading ή τουλάχιστον να μπορεί ο χρήστης να μοιράσει την διαδικασία στα threads της κάρτας γραφικών.

Ένα μειονέκτημα επίσης που παρατηρήθηκε κατά την εκτέλεση του αλγορίθμου στην κάρτα γραφικών είναι ότι αν το μέγεθος του dataframe υπερβαίνει το μέγεθος που διαθέτει η video ram η εκτέλεση του αλγορίθμου εμφανίζει πρόβλημα. Ενώ σε περιβάλλον Apache Spark αν εξαντληθεί ο χώρος που παρέχεται από τις ram συνεχίζει στην δέσμευση χώρου στον σκληρό δίσκο με προφανές κόστος την μειωμένη ταχύτητα του σκληρού δίσκου σε σχέση με τις ram. Όμως το βασικό πρόβλημα λύνεται ο αλγόριθμος εμφανίζει αποτελέσματα. Βεβαίως και το Apache Spark θα εμφανίσει πρόβλημα αν εξαντληθεί και ο χώρος του σκληρού δίσκου όμως οι πιθανότητες αυτές είναι πολύ λίγες και το κόστος ανά gigabyte σε επίπεδο σκληρού δίσκου είναι πολύ μικρότερο σε σχέση με αυτό στις ram και ακόμα περισσότερο στην κάρτα γραφικών.

Για την αντιμετώπιση του προβλήματος της περιορισμένης μνήμης στην κάρτα γραφικών δημιουργήθηκαν dataset όπου να υποστηρίζονται από την κάρτα γραφικών και τα ίδια dataset εκτελέστηκαν και σε περιβάλλον Apache Spark.

Για την δημιουργία ενός cluster από κάρτες γραφικών σε σχέση με την δημιουργία cluster από υπολογιστές, είναι μικρότερο το κόστος σε έναν υπολογιστή να τοποθετηθούν τρεις κάρτες γραφικών σε σχέση με την δημιουργία δικτύου από τρεις υπολογιστές. Δεδομένου ότι σε καμία από τις δύο περιπτώσεις δεν γίνει μονόπλευρα η επιλογή πιο ακριβών προϊόντων με πιο αυξημένες επιδόσεις.

5 Κεφάλαιο – Διαδικτυακή Πλατφόρμα

Με σκοπό την εκτέλεση αλγορίθμων και την οπτικοποίηση των αποτελεσμάτων τους που δημιουργήθηκαν κατασκευάστηκε μια διαδικτυακή πλατφόρμα οπτικοποιήσης δεδομένων χωροκειμενικής σύζευξης. Συνοπτικά η πλατφόρμα υποστηρίζει τις εξής λειτουργείες:

- **Σύνολα Δεδομένων:** Ανέβασμα των δεδομένων που θα απαιτηθούν για την εκτέλεση του αλγορίθμου. Οπτικοποίηση μέρος του dataset είτε ολόκληρου του dataset. Επίσης διαγραφή ή λήψη dataset που υπάρχουν έχουν ανέβει ήδη στην πλατφόρμα.
- **Αλγόριθμοι:** Επιλογή .py αρχείου που περιλαμβάνει τον αλγόριθμο που θα εκτελεστεί στην πλατφόρμα. Δήλωση των παραμέτρων που δέχεται σαν είσοδο ο αλγόριθμος. Προβολή ενός πίνακα που δείχνει τις παραμέτρους που δέχεται κάθε αλγόριθμος από αυτές που έχουν ήδη ανέβει.
- **Εκτέλεση Αλγορίθμου:** Δίνεται η δυνατότητα της επιλογής του αλγορίθμου προς εκτέλεση και ο ορισμός των παραμέτρων που δέχεται σαν είσοδο ο αλγόριθμος και κατόπιν η επιλογή της εκτέλεσης του αλγορίθμου.
- **Αποτελέσματα:** Μετά την εκτέλεση του κάθε αλγορίθμου παράγεται ένα dataset που περιέχει τα αποτελέσματα της εκτέλεσης του αλγορίθμου και υπάρχει η δυνατότητα προβολής των αποτελεσμάτων σε χάρτη αυτού του dataset αλλά και οποιοδήποτε dataset αποτελεσμάτων έχει από αλγορίθμους που έχουν εκτελεστεί. Βεβαίως για να μπορούν να οπτικοποιηθούν τα αποτελέσματα θα πρέπει να συμφωνούν σε ένα προκαθορισμένο σχήμα των δεδομένων.

Οι αλγόριθμοι που έχει ανεβάσει ο χρήστης στην πλατφόρμα εκτελούνται σε περιβάλλον παράλληλης επεξεργασίας Apache Spark διαφοροποιώντας την παραπάνω πλατφόρμα από μια απλή εφαρμογή. Δηλαδή είναι αναγκαία η παροχή δικτύου από υπολογιστές και η εγκατάσταση του Apache Spark μέσω αυτής της διαφοράς δίνεται η δυνατότητα να αποκαλείται πλατφόρμα.

5.1 Τεχνολογίες

Για την διεκπεραίωση της παραπάνω πλατφόρμας χρησιμοποιήθηκαν οι εξής τεχνολογίες και βιβλιοθήκες:

- Flask(για την υλοποίηση του back end προγραμματισμού και την υλοποιήσω των APIs)
- HTML(για την υλοποίηση του frond end προγραμματισμού) API
- Plotly(για την προβολή των δεδομένων σε χάρτη)
- Pyspark(την υποστήριξη της εκτέλεσης των αλγορίθμων που έχουν γραφεί σε Python Spark)

Αναλυτικά οι παραπάνω τεχνολογίες αναφέρονται στο Παράρτημα Β.

5.2 Διαδικτυακή Πλατφόρμα

Στη συνέχεια θα παραθέσουμε screenshot με τις λειτουργίες που παρέχει η διαδικτυακή πλατφόρμα και περεταίρω επεξήγηση του τρόπου με τον οποίον υλοποιήθηκαν οι συγκεκριμένες λειτουργίες.

Όταν κάποιος επισκεφθεί την ιστοσελίδα θα αντικρίσει ένα μήνυμα καλωσορίσματος, μια πρόταση που περιγράφει τον στόχο της σελίδας και μια navigation bar για να περιηγηθεί στην ιστοσελίδα όπως φαίνεται στην παρακάτω εικόνα.

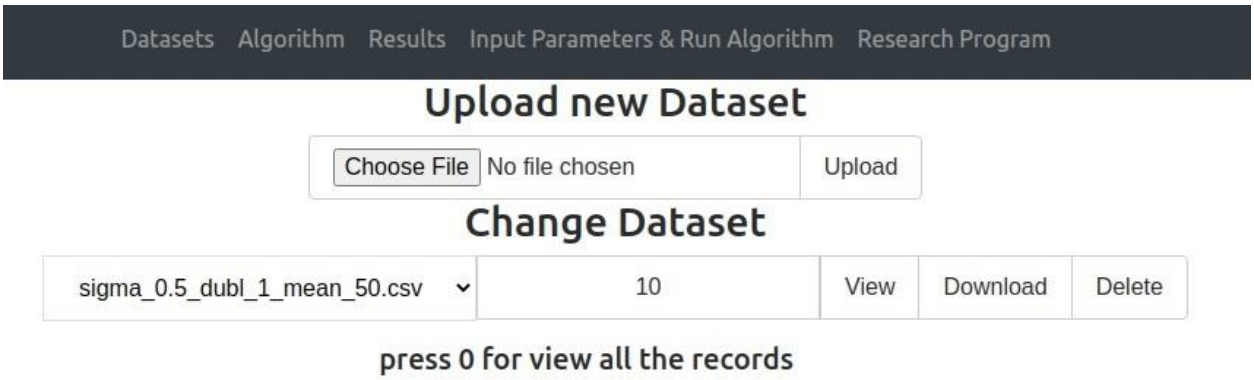
Εικόνα 21 Διαδικτυακή Πλατφόρμα Κεντρική Σελίδα

Η navigation bar διαθέτει πέντε διαφορετικά tabs.

- **Datasets:** Μπορεί ο χρήστης να ανεβάσει κάποιο dataset και να το οπτικοποιήσει σε χάρτη
- **Algorithm:** Ο χρήστης μπορεί να ανεβάσει κάποιο αλγόριθμο και να δηλώσει τις παραμέτρους που δέχεται σαν είσοδο ο αλγόριθμος. Επίσης, μπορεί να δει τις παραμέτρους των αλγορίθμων που υπάρχουν ήδη ανεβασμένοι.
- **Results:** Ο χρήστης μπορεί να προβάλει τα αποτελέσματα από την εκτέλεση του αλγορίθμου σε χάρτη.
- **Input Parameter & Run Algorithm:** Ο χρήστης ορίζει τις παραμέτρους που δέχεται ο αλγόριθμος και κατόπιν και εκτελεί τον αλγόριθμο.
- **Research Program:** Οδηγεί τον χρήστη στην σελίδα του ερευνητικού προγράμματος στα πλαίσια του οποίου δημιουργήθηκε η πλατφόρμα.

5.2.1 Σύνολα Δεδομένων

Στο tab dataset ο χρήστης θα δει το περιεχόμενο της παρακάτω εικόνας.



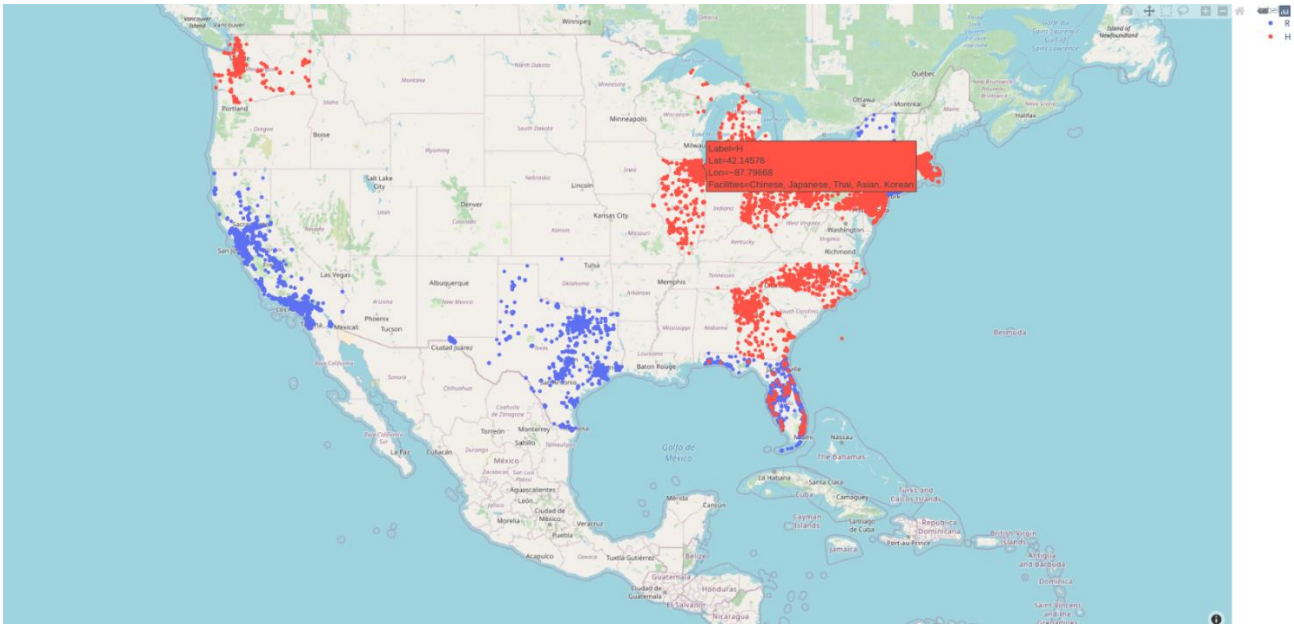
Εικόνα 22 Διαδικτυακή Πλατφόρμα Σύνολα Δεδομένων

Πατώντας το κουμπί “Choose File” ανοίγει στον χρήστη ένα νέο παράθυρο όπου περιέχονται όλα τα αρχεία του υπολογιστή του και μπορεί να επιλέξει το αρχείο που θέλει να ανεβάσει. Η διαδικασία του

upload τελειώνει εφόσον ο χρήστης πιέσει το κουμπί Upload, το αρχείο ανεβαίνει και αποθηκεύεται σε έναν φάκελο μες στον server που έχει προκαθοριστεί και λέγεται “upload_dataset”. Επίσης, εφόσον τελειώσει το ανέβασμα του αρχείου, εμφανίζεται κατάλληλο μήνυμα ενημέρωσης που επιβεβαιώνει αν ανέβηκε το αρχείο.

Στην ίδια σελίδα μπορεί ο χρήστης να επιλέξει κάποιο από τα ήδη υπάρχοντα dataset, να συμπληρώσει το πλήθος των εγγραφών που θέλει να προβληθούν από το συγκεκριμένο dataset και πατώντας το κουμπί View να προβληθεί το dataset σε έναν χάρτη σε νέο tab μέσω της βιβλιοθήκης plotly. Βέβαια όπως αναφέρει και στην ένδειξη στην εικόνα συμπληρώνοντας τον αριθμό 0 γίνεται προβολή όλου του dataset. Με το κουμπί Download ο χρήστης μπορεί να κατεβάσει ένα από τα ήδη ανεβασμένα dataset που υπάρχουν στην πλατφόρμα και με το κουμπί Delete διαγράφει όποιο dataset επιλέξει.

Όταν ο χρήστης επιλέξει να κάνει προβολή του dataset ο τύπος προβολής θα είναι ο εξής



Εικόνα 23 Διαδικτυακή Πλατφόρμα Οπτικοποίηση Συνόλου Δεδομένων

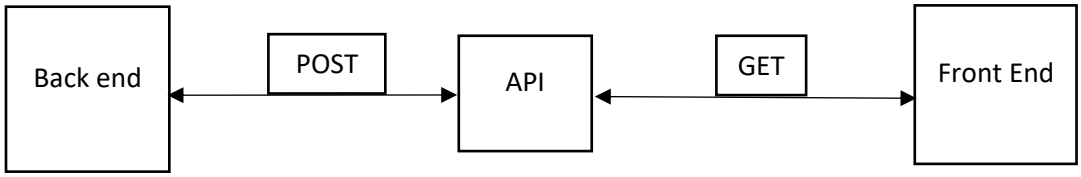
Τα μπλε και τα κόκκινα είναι τα δύο διαφορετικά είδη αντικειμένων που περιλαμβάνει το dataset για παράδειγμα ξενοδοχεία και εστιατόρια. Επάνω δεξιά αναγράφεται τι συμβολίζουν τα μπλε και τι τα κόκκινα σύμφωνα με την μεταβλητή label του dataset. Επίσης, πηγαίνοντας το ποντίκι πάνω σε κάθε σημείο παρατηρούμε ότι υπάρχει ένα αναδυόμενο παράθυρο που αναφέρει τα χαρακτηριστικά του κάθε σημείου όπως φαίνεται στην παρακάτω εικόνα.

Εικόνα 24 Διαδικτυακή Πλατφόρμα Ετικέτες Σημείων

Για να γίνει προβολή του dataset θα πρέπει να είναι αποθηκευμένο σε csv αρχείο και να έχει το παρακάτω σχήμα

Ονόματα Μεταβλητών	**Δεδομένα**
Name	Όνομα του αντικειμένου για παράδειγμα όνομα ξενοδοχείου
Lon	Γεωγραφικό μήκος του αντικειμένου.
Lat	Γεωγραφικό πλάτος του αντικειμένου.
Label	Label για το τι είδος αντικείμενο είναι για παράδειγμα ξενοδοχείο ή εστιατόριο
Facilities	Κειμενική πληροφορία για κάθε αντικείμενο

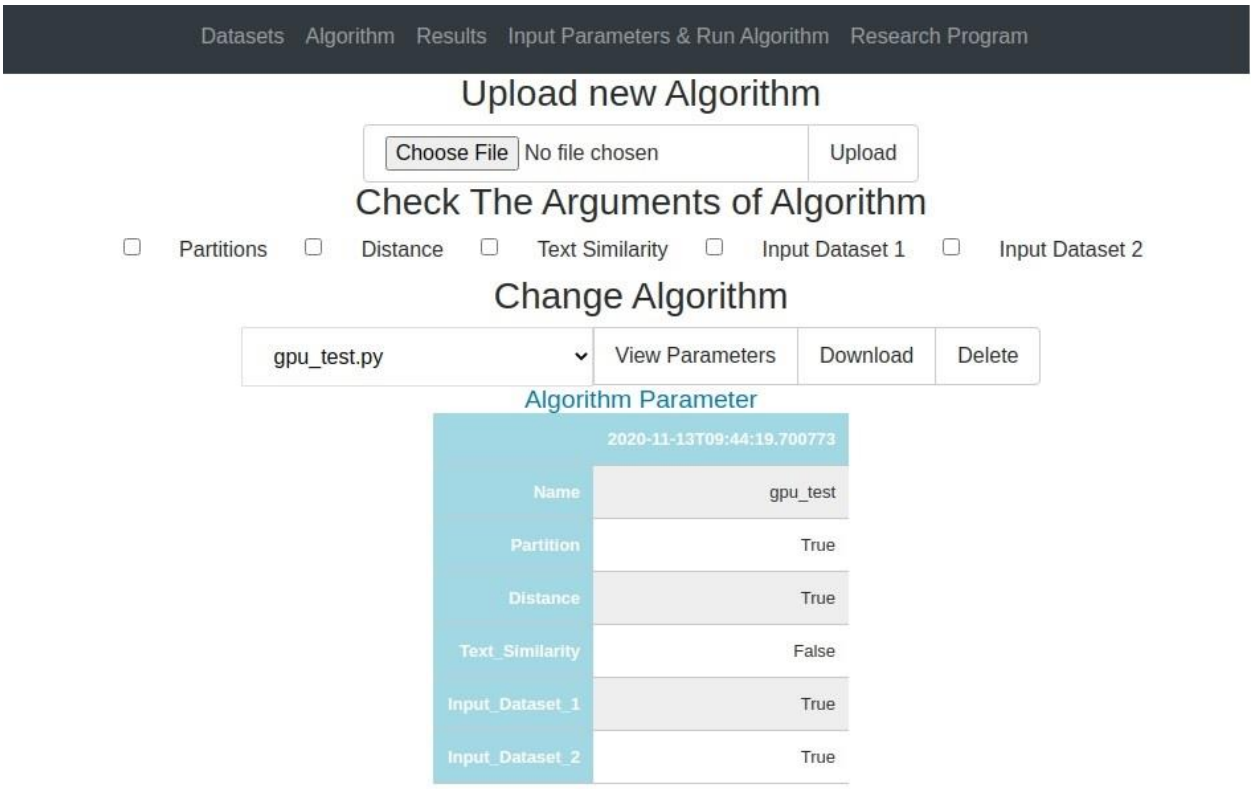
Για την υλοποίηση των παραπάνω λειτουργειών χρησιμοποιήθηκε μια μέθοδο που περιέχει όλες τις αναφερόμενες λειτουργείες και όλα τα APIs είναι τις παρακάτω μορφής. Για κάθε λειτουργεία υπάρχει ένα API.



Για παράδειγμα ο αριθμός εγγραφών του dataset που επιθυμεί ο χρήστης να εμφανιστούν είναι μια τιμή k που εκχωρείται στο front end. Για να εκτελεστεί η δημιουργία του χάρτη η τιμή αυτή γίνεται get από το front end και post στο back end για να ξεκινήσει η δημιουργία του χάρτη κάνοντας ανάγνωση από το σύνολο δεδομένων τις k πρώτες εγγραφές που έχει δηλώσει ο χρήστης.

5.2.2 Αλγόριθμοι

Αφού ο χρήστης ανέβασε το σύνολο δεδομένων που επιθυμεί και το οπτικοποίησε, πρέπει να ανεβάσει και τον αλγόριθμο που επιθυμεί να εκτελέσει στην πλατφόρμα. Επισυνάπτεται ένα screenshot με τις λειτουργίες που παρέχονται στο tab Algorithm.



Εικόνα 25 Διαδικτυακή Πλατφόρμα Ανέβασμα Νέου Αλγορίθμου

Κάτω από την επικεφαλίδα από την επικεφαλίδα με το όνομα “Upload new Algorithm” παρατηρούμε ότι είναι ακριβώς η ίδια φόρμα με αυτή στο tab dataset. Δηλαδή πατά ο χρήστης το κουμπί “Choose File” επιλέγει το αρχείο που θέλει και το ανεβάζει. Βέβαια στην συγκεκριμένη περίπτωση έχουν προστεθεί 2 ακόμα λειτουργίες. Γίνεται ένας έλεγχος ώστε το αρχείο που προσπαθεί να ανεβάσει ο χρήστης να είναι .py γιατί οποιοσδήποτε άλλος τύπος αρχείου είναι αδύνατον να εκτελεστεί σε αυτή την πλατφόρμα. Δεύτερον, πριν πατήσει το κουμπί “Upload” ο χρήστης θα πρέπει να επιλέξει τις παραμέτρους που δέχεται σαν είσοδο ο αλγόριθμος που σκοπεύει να ανεβάσει. Αν δεν επιλέξει καμία παράμετρο αυτές παίρνουν όλες την τιμή False. Οι παράμετροι αποθηκεύονται σε ένα αρχείο .csv όπου μαζί με αυτές αποθηκεύεται και η τρέχουσα ημερομηνία και ώρα που έγινε η καταχώριση. Έτσι, αν κάποιος χρήστης ανεβάσει το ίδιο αρχείο πολλές φορές, η ημερομηνία και ώρα που μετράται είναι η πιο πρόσφατη. Επιπλέον τα αρχεία των αλγορίθμων αποθηκεύονται σε ένα φάκελο στο server που λέγεται “Upload Script”.

Κάτω από την επικεφαλίδα “Change Algorithm” ο χρήστης μπορεί να επιλέξει τον αλγόριθμο που επιθυμεί και να δει τις παραμέτρους που έχουν δηλωθεί. Επίσης μπορεί να κατεβάσει κάποιον από τους ήδη υπάρχοντες αλγορίθμους μέσω του κουμπιού “Download”. Όπως επίσης και να διαγράψει κάποιον μέσω του κουμπιού “Delete”.

5.2.3 Αποτελέσματα

Αφού εκτελεστεί ο αλγόριθμος, παράγεται ένα dataset του οποίου το όνομα το δηλώνει ο χρήστης και το αποθηκεύει στον φάκελο του server που ονομάζεται “results”. Το περιεχόμενο του tab Result προβάλλεται στην παρακάτω εικόνα.

Εικόνα 26 Διαδικτυακή Πλατφόρμα Αποτελέσματα

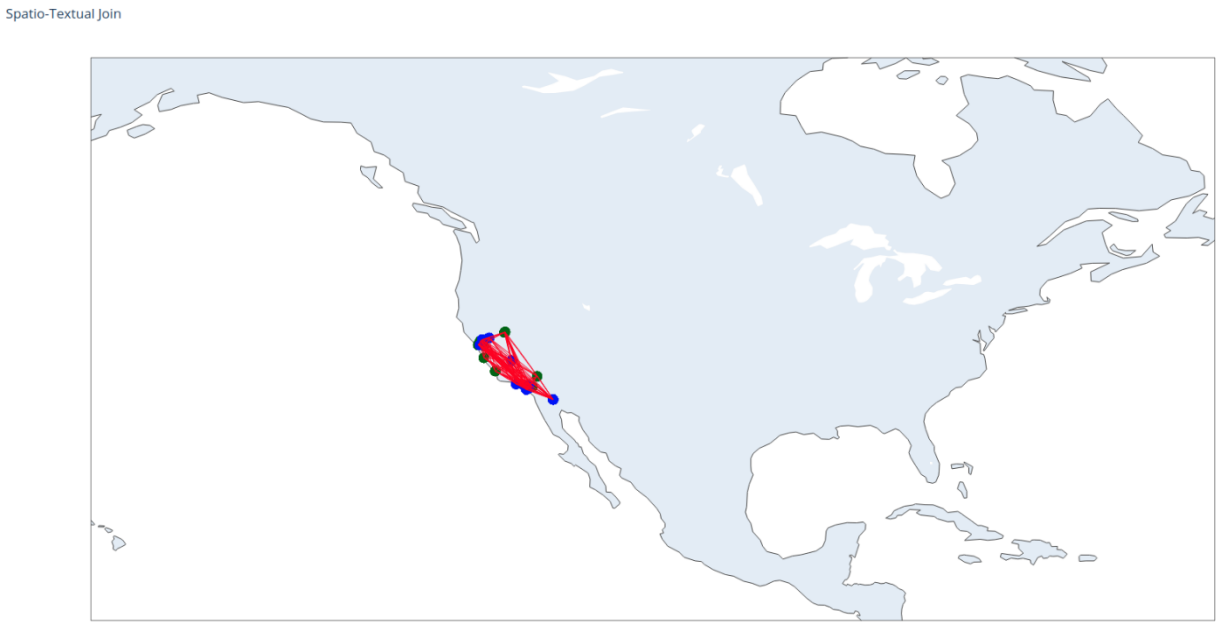
Ο χρήστης μπορεί να επιλέξει ένα από τα dataset με τα αποτελέσματα που έχουν παραχθεί είτε από κάποιον αλγόριθμο που εκτέλεσε είτε από παλιότερους αλγορίθμους να επιλέξει το πλήθος των εγγραφών που επιθυμεί να προβάλει και να το προβάλει σε χάρτη. Στην περίπτωση που πατήσει μηδέν όπως αναφέρει και στο αντίστοιχο μήνυμα προβάλλεται όλο το dataset. Αυτή η λειτουργία εκτελείται αφού πατήσει το κουμπί “Visualize”. Αν πατήσει το κουμπί “Download” μπορεί να κατεβάσει το επιλεγμένο dataset και με το κουμπί “Delete” να το διαγράψει.

Για να γίνει η προβολή των αποτελεσμάτων σε χάρτη τα δεδομένα πρέπει να είναι αποθηκευμένα σε μορφή .csv και να ακολουθούν το παρακάτω σχήμα.

Ονόματα Μεταβλητών	**Δεδομένα**
name1	Όνομα του πρώτου είδους αντικειμένου για παράδειγμα όνομα ξενοδοχείου
lon1	Γεωγραφικό μήκος του πρώτου είδους αντικειμένου.
lat1	Γεωγραφικό πλάτος του πρώτου είδους αντικειμένου.
label_1	Label του πρώτου είδους αντικειμένου
facilities1	Κειμενική πληροφορία για του πρώτου είδους αντικείμενα
name2	Όνομα του δεύτερο είδους αντικειμένου για παράδειγμα όνομα εστιατορίου
lon2	Γεωγραφικό μήκος του δεύτερου είδους αντικειμένου.
lat2	Γεωγραφικό πλάτος του δεύτερου είδους αντικειμένου.
label_2	Label του δεύτερου είδους αντικειμένου
facilities2	Κειμενική πληροφορία για τα δεύτερου είδους αντικείμενα

Πίνακας 13 Σχήμα των Χωρο-κειμενικών Συνόλων Δεδομένων

Ο χάρτης που προβάλλει τα dataset με τα αποτελέσματα έχει τον παρακάτω τύπο



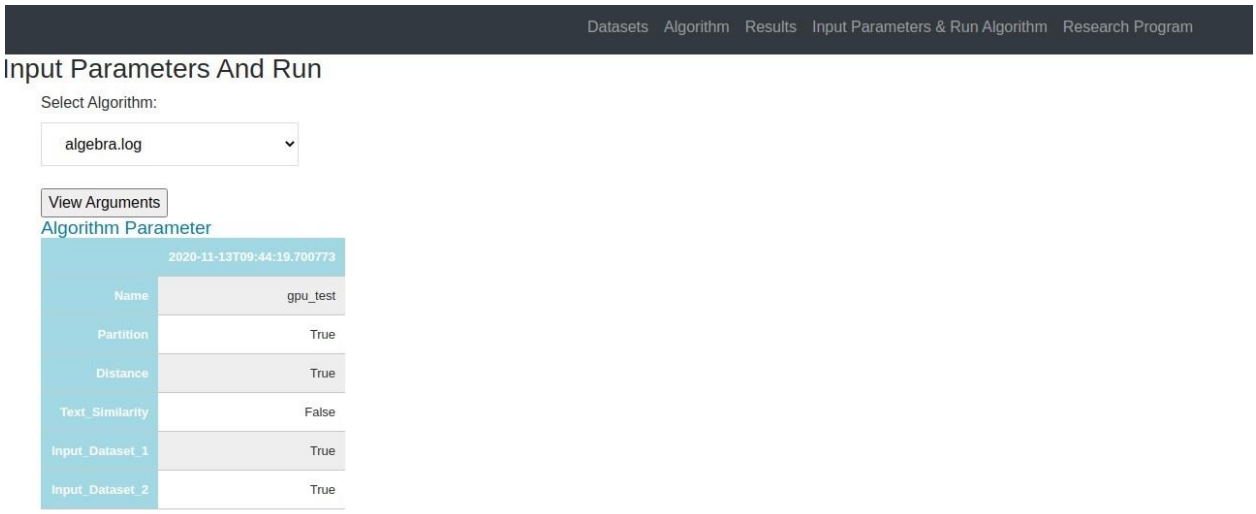
Εικόνα 27 Διαδικτυακή Πλατφόρμα Οπτικοποίηση Αποτελεσμάτων

Με μπλε εμφανίζονται τα αντικείμενα του ενός είδους για παράδειγμα ξενοδοχεία. Με πράσινο τα αντικείμενα του άλλου είδους για παράδειγμα εστιατόρια και οι κόκκινες γραμμές δείχνουν ποια ξενοδοχεία συνδέονται με ποια εστιατόρια.

5.2.4 Εκτέλεση Αλγορίθμου

Στο συγκεκριμένο tab μπορεί ο χρήστης να εκτελέσει όποιον αλγόριθμο επιθυμεί από αυτούς που ήδη έχουν ανέβει και με όποιες παραμέτρους θέλει από αυτές που δέχεται σαν είσοδο.

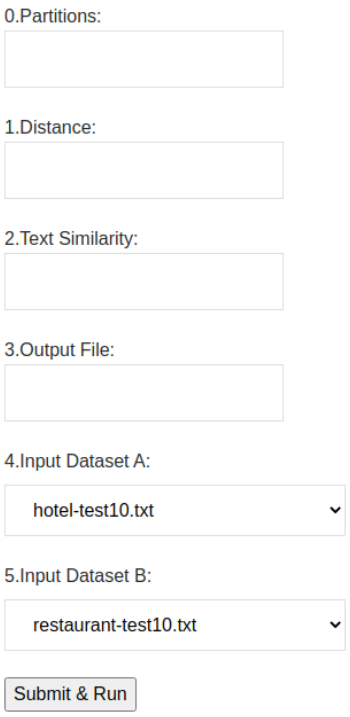
Για διευκόλυνση του χρήστη μπορεί και σε αυτό το tab να δει τις παραμέτρους που απαιτεί κάθε αλγόριθμος για να εκτελεστεί. Όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 28 Διαδικτυακή Πλατφόρμα Εκτέλεση Αλγορίθμου 1

Επιλέγοντας κάποιον από τους αλγορίθμους που έχουν ανέβει και πατώντας το κουμπί “View Arguments” προβάλλεται σε πίνακα το όνομα του αλγορίθμου και το ποιες παραμέτρους χρειάζεται σαν είσοδο ο αλγόριθμος για να εκτελεστεί.

Στην συνέχεια υπάρχει μια φόρμα όπως φαίνεται στην παρακάτω εικόνα για την εκτέλεση του αλγορίθμου.



Εικόνα 29 Διαδικτυακή Πλατφόρμα Εκτέλεση Αλγορίθμου 2

Οι παραπάνω παράμετροι δηλώνουν τα εξής:

- 0.Partitions: Αριθμός των partitions
- 1.Distance: Το όριο της απόστασης των δύο ειδών αντικειμένων
- 2.Text Similarity: Το όριο της κειμενικής ομοιότητας των δύο ειδών αντικειμένων
- 3.Output File: Τον τίτλο του αρχείου με τα αποτελέσματα που θα εξάγει ο αλγόριθμος αφού εκτελεστεί.
- 4.Input Dataset 1: Επιλογή ενός από τα dataset που έχουν ήδη ανέβει στην πλατφόρμα.
- 5.Input Dataset 2: Επιλογή του δεύτερου dataset εφόσον ο αλγόριθμος απαιτεί δύο dataset για την εκτέλεσή του.

Πριν ο χρήστης ανεβάσει κάποιον αλγόριθμο για να είναι ο αλγόριθμος πλήρως λειτουργικός με την πλατφόρμα θα πρέπει να εισάγει τα εξής:

```
print("partitions",sys.argv[0])
print("distance",sys.argv[1])
print("text_similarity",sys.argv[2])
print("Output_File",sys.argv[3])
print("dataset_path 4",sys.argv[4])
print("dataset_path 5",sys.argv[5])
```

Εικόνα 30 Διαδικτυακή Πλατφόρμα Εισαγωγή Μεταβλητών

Οι παραπάνω μεταβλητές sys.argv[] είναι οι παράμετροι για την εκτέλεση του αλγορίθμου οι οποίοι εισάγονται από το Flask στο python script. Για την εισαγωγή αυτών των παραμέτρων απαιτείται η εισαγωγή της βιβλιοθήκης import sys. Στη συνέχεια οι παράμετροι ακολουθούν την αντιστοίχιση σύμφωνα με την παραπάνω εικόνα.

Για το διάβασμα κάποιου dataset είτε την εγγραφή του απαιτείται η χρήση της εντολής: os.path.join(sys.argv[3]) όταν επρόκειτο να χρησιμοποιηθεί για να γράψει κάποιο dataset και os.path.join(sys.argv[4]), os.path.join(sys.argv[5]) όταν επρόκειτο να διαβάσει κάποιο dataset.Για παράδειγμα η εντολή διαβάσματος ενός csv αρχείου πρέπει να είναι κάπως έτσι: spark.read.csv(os.path.join(sys.argv[4])).

Η αντιστοιχία των 0-5 στα sys.argv[] φαίνεται στην σελίδα στο tab “Input Parameters & Run Algorithm” όπου πριν από κάθε παράμετρο φαίνεται ο αριθμός του sys.argv[]. Δεν είναι δεσμευτικό οι μεταβλητές των παραμέτρων να χρησιμοποιηθούν σε κάποιον αλγόριθμο με τα ονόματα που έχουν οριστεί. Δηλαδή μπορεί ένας χρήστης το sys.argv[1] που είναι το Distance να το χρησιμοποιήσει για να εισάγει ένα νούμερο στον αλγόριθμό του με μια άλλη λειτουργία. Βέβαια αυτό που είναι δεσμευτικό είναι η χρήση τους να συμφωνεί με τον τύπο μεταβλητής που έχει δηλωθεί. Όπως φαίνεται στον παρακάτω πίνακα.

Παράμετροι	Τύπος Μεταβλητής
sys.argv[0]	Integer
sys.argv[1]	Float
sys.argv[2]	Float
sys.argv[3]	String
sys.argv[4]	String
sys.argv[5]	String

Πίνακας 14 Διαδικτυακή Πλατφόρμα Εισαγωγή Μεταβλητών

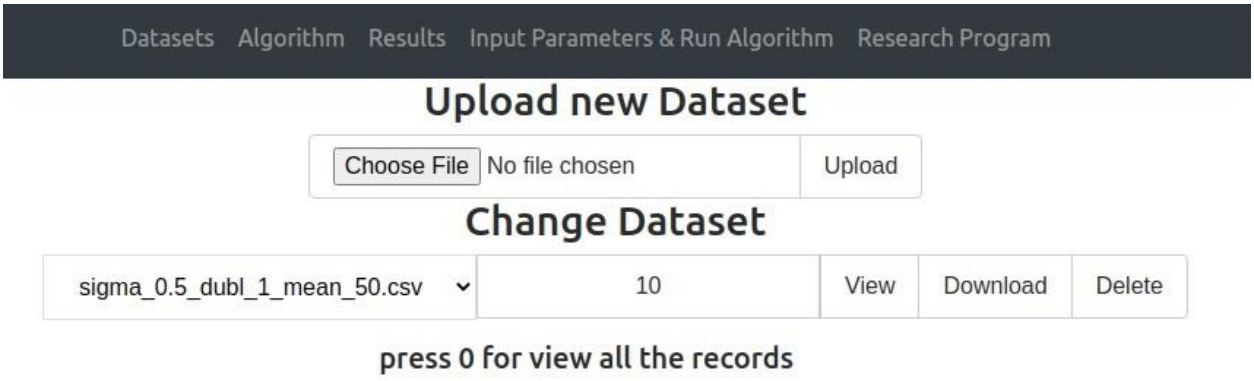
5.3 Επεκτασιμότητα

Ο σκοπός δημιουργίας της πλατφόρμας είναι η εύκολη επεκτασιμότητα δηλαδή να μπορεί ο χρήστης να ανεβάζει τα σύνολα δεδομένων του και τον αλγόριθμό του, να τον εκτελεί και να κάνει οπτικοποίηση των αποτελεσμάτων σε χάρτη. Βέβαια μπορεί να επιλέξει να έχει σαν είσοδο ο αλγόριθμός του ένα από τα σύνολα δεδομένων που υπάρχουν ήδη διαθέσιμα στην πλατφόρμα. Επίσης, έχει την δυνατότητα να εκτελέσει έναν από τους αλγόριθμους που υπάρχουν ήδη στην πλατφόρμα με σύνολα δεδομένων δικά του ή από αυτά που υπάρχουν ήδη στην πλατφόρμα.

Η διαδικασία που πρέπει να ακολουθήσει ο χρήστης για να εκτελέσει των δικό του αλγόριθμο με τα δικά του σύνολα δεδομένων είναι η εξής:

1. Ανέβασμα των συνόλων δεδομένων στην πλατφόρμα

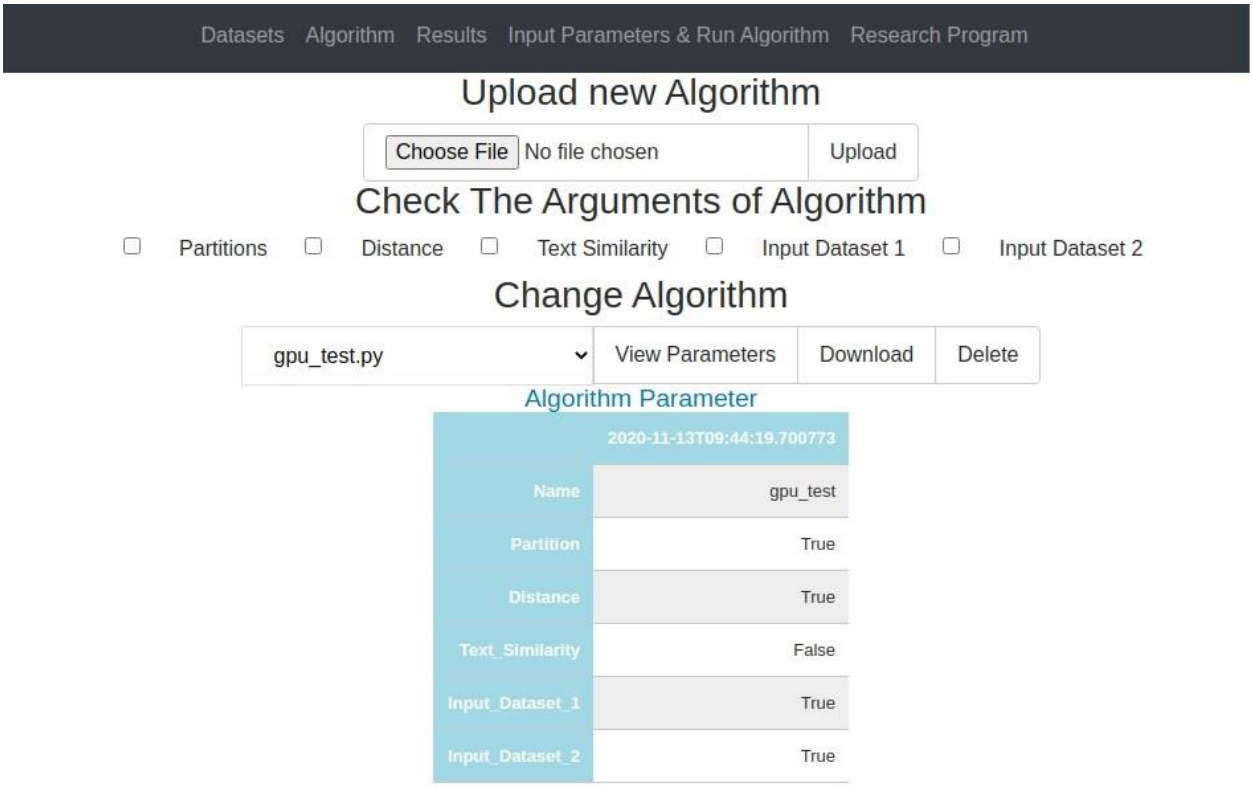
Πηγαίνοντας στο tab dataset μπορεί ο χρήστης να επιλέξει ένα ή περισσότερα σύνολα δεδομένων να ανεβάσει από τον υπολογιστή του. Επιπλέον, στο ίδιο tab έχει την δυνατότητα να οπτικοποιήσει μέρος του συνόλου δεδομένων ή ολόκληρο το σύνολο δεδομένων. Για την οπτικοποίηση ολόκληρου του συνόλου δεδομένων, ο χρήστης συμπληρώνει την τιμή 0, αλλιώς οποιαδήποτε άλλη τιμή αντικατοπτρίζει το πλήθος των πρώτων εγγραφών που θα οπτικοποιηθούν.



Εικόνα 31 Διαδικτυακή Πλατφόρμα Σύνολα Δεδομένων

2. Ανέβασμα του αλγορίθμου

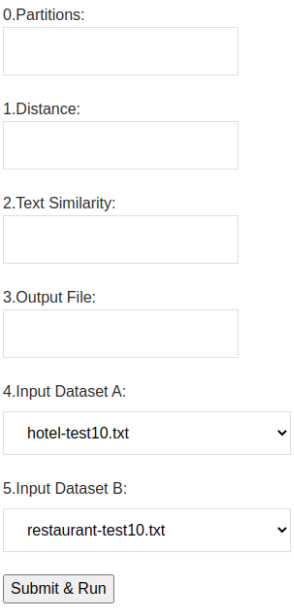
Στη συνέχεια ο χρήστης πρέπει να ανατρέξει στην καρτέλα Algorithm όπου θα ανεβάσει τον αλγόριθμο που έχει δημιουργήσει και θα επιλέξει τις παραμέτρους που δέχεται σαν όρισμα ο αλγόριθμος του.



Εικόνα 32 Διαδικτυακή Πλατφόρμα Ανέβασμα Νέου Αλγορίθμου

3. Εκτέλεση του αλγορίθμου

Ο χρήστης ανατρέχει στο tab Input Parameter & Run Algorithm για την εκτέλεση του αλγορίθμου όπου εκχωρεί τις παραμέτρους που δέχεται σαν είσοδο ο αλγόριθμος και τα σύνολα ή σύνολο δεδομένων που θα εκτελεστεί ο αλγόριθμος.

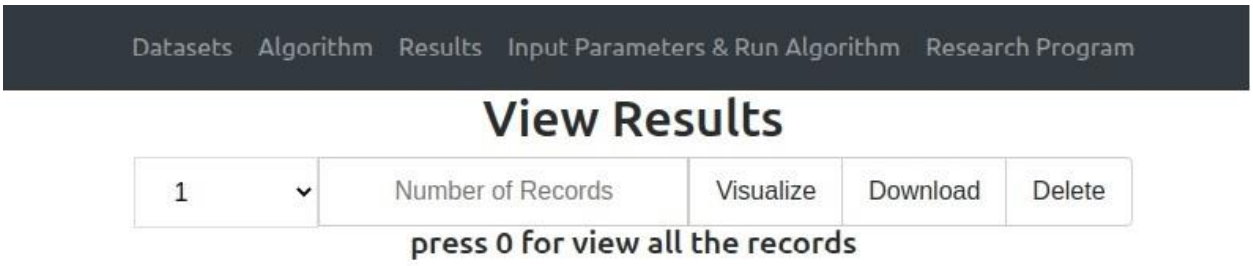


Εικόνα 33 Διαδικτυακή Πλατφόρμα Εκτέλεση Αλγορίθμου

Στην συνέχεια με την επιλογή του κουμπιού Submit & Run ο αλγόριθμος εκτελείται.

4. Προβολή Αποτελεσμάτων

Στο tab Result ο χρήστης επιλέγει το σύνολο δεδομένων που παράγει σαν αποτέλεσμα ο αλγόριθμος που εκτελέστηκε και έχει την δυνατότητα να το κάνει οπτικοποίηση και να το κατεβάσει στον υπολογιστή του.



Εικόνα 34 Διαδικτυακή Πλατφόρμα Αποτελέσματα

Η πλατφόρμα έχει την δυνατότητα να εκτελέσει οποιοδήποτε αλγόριθμο χρησιμοποιεί pyspark και ο χρήστης να ανεβάσει οτιδήποτε τύπο δεδομένων και να παράξει ο αλγόριθμος επίσης οποιοδήποτε τύπο δεδομένων. Βέβαια για την οπτικοποίηση του συνόλων δεδομένων είτε των αποτελεσμάτων, τα δεδομένα πρέπει να ακολουθούν ένα συγκεκριμένο σχήμα και να είναι αρχεία csv.

6 Κεφάλαιο - Συμπεράσματα και Μελλοντική Εργασία

6.1 Συμπεράσματα

Η εκτέλεση της σύζευξης των χωρικών και χωρο-κειμενικών δεδομένων είναι μια ακριβή υπολογιστική πράξη που δυσχεραίνει την ταχεία επεξεργασία τους. Επίσης, το πρόβλημα κλιμακώνεται όταν τα δεδομένα είναι μεγάλου όγκου και πρέπει να βρεθεί η χωρική απόσταση των σημείων τους και η κειμενική ομοιότητα αυτών. Για τους παραπάνω λόγους κρίνεται απαραίτητη η παράλληλη επεξεργασία τους σε κατανεμημένα συστήματα, δηλαδή σε συστήματα που αποτελούνται από παραπάνω από έναν υπολογιστές. Ένα σημαντικό πρόβλημα για την παράλληλη επεξεργασία των δεδομένων αποτελεί η αποδοτική διαμέριση (partitioning) αυτών. Η διαμέριση των δεδομένων είναι πολύ σημαντική διότι πρέπει να επιμερίζεται ισάξια ο φόρτος εργασίας στους υπολογιστές, ώστε να επιτυγχάνεται η βέλτιστη απόδοση του αλγορίθμου. Βέβαια, ακόμα ένα σημαντικό πρόβλημα στην διαμέριση των δεδομένων είναι κατά την διαδικασία της διαμέρισης να μην χαθούν εγγραφές από τα δεδομένα ή σύνολα εγγραφών που είναι πιθανό να παράγουν σημαντική πληροφορία.

Για την χρονομέτρηση του χρόνου εκτέλεσης ενός αλγορίθμου που έχει υλοποιηθεί σε περιβάλλον Apache Spark δεν αρκεί να μετρηθεί ο χρόνος εκτέλεσης του αλγορίθμου διότι οι εντολές στο Apache Spark δεν εκτελούνται σειριακά και η σειρά εκτέλεσης δεν είναι γνωστή στον χρήστη. Ένας ενδεικτικός τρόπος μέτρησης είναι να χρονομετρηθεί μέσα από το περιβάλλον Apache Spark μια εντολή που για να παράγει αποτέλεσμα να πρέπει να προσπελαστούν όλες οι εγγραφές, όπως για παράδειγμα η count(). Έτσι διασφαλίζεται ότι έχουν εκτελεστεί όλες οι προηγούμενες εντολές και είναι ένα ρεαλιστικό κριτήριο για την σύγκριση αλγορίθμων.

Παρατηρήθηκε ότι στην περίπτωση του partitioning σε cross join query η μείωση των εγγραφών είναι τόσο μεγάλη που επιβάλλεται να κάνεις partitioning. Επιπλέον, μείωση των εγγραφών σημαίνει και μείωση του χρόνου εκτέλεσης.

Παρατηρήθηκε ότι κατά την αύξηση των partition από 4 σε 8 και 12 σύμφωνα με τους χρόνους του Spark υπάρχει μια μείωση των χρόνων στο μισό. Κατά την αύξηση των partition από 12 σε 16 οι χρόνοι εκτέλεσης παραμένουν ίδιοι και αυτό οφείλεται στο γεγονός ότι ο master και ο slave έχουν σύνολο 12 threads, οπότε ο βέλτιστος χρόνος εκτέλεσης του προγράμματος προκύπτει όταν ο αριθμός των partitions είναι ο ίδιος με τον αριθμό των threads.

Στην περίπτωση του Apache Spark όπως και στην περίπτωση της GPU παρατηρήθηκε ότι, μετά από επαναλαμβανόμενες εκτελέσεις των αλγορίθμων η cache είτε του επεξεργαστή είτε της κάρτας γραφικών, κρατά ένα κομμάτι πληροφορίας που στην δεύτερη ή τρίτη εκτέλεση έχει σαν αποτέλεσμα την μείωση του χρόνου εκτέλεσης. Άρα έχει σαν επακόλουθο την αλλοίωση των αποτελεσμάτων. Επομένως, σε οποιαδήποτε δοκιμή γίνεται είτε στην κάρτα γραφικών είτε στον επεξεργαστή, πρέπει να λαμβάνετε υπόψη ο καθαρισμός της μνήμης cache.

Κατά την διαδικασία της εκτέλεσης πειραμάτων σε περιβάλλον Apache Spark παρατηρήθηκε ότι αν δηλωθούν περισσότερα partitions από τα threads που διαθέτει ο επεξεργαστής ή οι επεξεργαστές, ο χρόνος παύει να μειώνεται οπότε μας οδηγεί στο συμπέρασμα ότι το Apache Spark κάνει κάποιου είδους threading. Σε αντίθεση με την κάρτα γραφικών όπου παρότι έγιναν προσπάθειες να γίνει partitioning στα

δεδομένα επειδή δεν υπήρχε διαθέσιμη δεύτερη κάρτα δεν υποστηριζόταν το partitioning άρα δεν υπάρχει η δυνατότητα να γίνει κάποιου είδους threading ή τουλάχιστον να μπορεί ο χρήστης να μοιράσει την διαδικασία στα threads της κάρτα γραφικών.

Ένα μειονέκτημα επίσης που παρατηρήθηκε κατά την εκτέλεση του αλγορίθμου στην κάρτα γραφικών είναι ότι αν το μέγεθος του dataframe υπερβαίνει το μέγεθος που διαθέτει η video ram η εκτέλεση του αλγορίθμου εμφανίζει πρόβλημα. Ενώ σε περιβάλλον Apache Spark αν εξαντληθεί ο χώρος που παρέχεται από τις ram συνεχίζει στην δέσμευση χώρου στον σκληρό δίσκο με προφανές κόστος την μειωμένη ταχύτητα του σκληρού δίσκου σε σχέση με τις ram. Όμως το βασικό πρόβλημα λύνεται ο αλγόριθμος εμφανίζει αποτελέσματα. Βεβαίως και το Apache Spark θα εμφανίσει πρόβλημα αν εξαντληθεί και ο χώρος του σκληρού δίσκου, όμως οι πιθανότητες αυτές είναι πολύ λίγες και το κόστος ανά gigabyte σε επίπεδο σκληρού δίσκου είναι πολύ μικρότερο σε σχέση με αυτό στις ram και ακόμα περισσότερο στην κάρτα γραφικών.

Για την αντιμετώπιση του προβλήματος του χώρου στην κάρτα γραφικών δημιουργήθηκαν dataset που να υποστηρίζονται από την κάρτα γραφικών και τα ίδια dataset εκτελέστηκαν και σε περιβάλλον Apache Spark.

Για την δημιουργία ενός cluster από κάρτες γραφικών σε σχέση με την δημιουργία cluster από υπολογιστές, είναι μικρότερο το κόστος σε έναν υπολογιστή να τοποθετηθούν τρεις κάρτες γραφικών σε σχέση με την δημιουργία δικτύου από τρεις υπολογιστές. Δεδομένου ότι σε καμία από τις δύο περιπτώσεις δεν γίνεται μονόπλευρα η επιλογή πιο ακριβών προϊόντων με πιο αυξημένες επιδόσεις.

6.2 Μελλοντική Εργασία

Για περαιτέρω πειραματική διαδικασία αξίζει η σύγκριση του ήδη υπάρχοντος αλγορίθμου με αλγορίθμους που υποστηρίζουν διαφορετικές τεχνικές διαμέρισης δεδομένων όπως διαμέρασης των δεδομένων με βάση τη χρήση της μεθοδολογίας ανάπτυξης δέντρου. Όπως επίσης και η διαμέριση των δεδομένων βάσει των κειμενικών τους κριτηρίων και όχι των χωρικών τους. Θα μπορούσε να εφαρμοστεί η ανάπτυξη ενός quad-tree σε συνδυασμό με την χρήση signature για κάθε λέξη(token) και τα signature να αποτελούν είσοδο μιας inverted list όπου μέσω αυτής θα λαμβάνονταν τα αποτελέσματα της χωρο-κειμενικής σύζευξης.

Για την εκτέλεση του αλγορίθμου χωρικής σύζευξης χρησιμοποιώντας πόρους της κάρτας γραφικών θα ήταν άξιο υλοποίηση η χρήση περισσότερων καρτών γραφικών πέραν της μίας για να μπορεί να εφαρμοστεί και κάποια μέθοδος partitioning διότι στην εκτέλεση του αλγορίθμου με την χρήση της blazingsql δεν υπάρχει η δυνατότητα του να γίνει partitioning με την χρήση μιας κάρτας. Έν αντιθέσει, το Apache Spark ακόμα και σε έναν υπολογιστή χρησιμοποιώντας αυτόνομα κάθε thread του επεξεργαστή έχει τη δυνατότητα του partitioning ακόμα και στην εκτέλεση αλγορίθμου σε έναν υπολογιστή.

Τέλος, όσο αφορά τη δημιουργία της διαδικτυακής πλατφόρμας, άξιο εργασίας για την τελειοποίηση αυτής είναι οποιονδήποτε pyspark αλγόριθμο ανεβάζει ο χρήστης να εκτελείται σε cluster υπολογιστών και εκτός από αποτελέσματα να λαμβάνει και τον χρόνο εκτέλεσης του αλγορίθμου.

7 Βιβλιογραφία

[1] Panagiotis Bouros, Shen Ge, Nikos Mamoulis:
Spatio-textual similarity joins. Proc. VLDB Endow. 6(1): 1-12 (2012)

[2] Yu Zhang, Youzhong Ma, Xiaofeng Meng:
Efficient Spatio-textual Similarity Join Using MapReduce. WI-IAT (2) 2014: 52-59

[3] Huiqi Hu, Guoliang Li, Zhifeng Bao, Jianhua Feng, Yongwei Wu, Zhiguo Gong, Yaoqiang Xu:Top-k Spatio-Textual Similarity Join. TKDE 2016, Volume 28 Issue 2, February 2016 Pages 551-565

[4] J. Ballesteros, A. Cary, and N. Rishe: Spsjoin: Parallel spatial similarity joins. In Proc. of the 19th ACM SIGSPATIAL GIS Conf., pages 481-484, 2011

[5] J. Rao, J. Lin, and H. Samet: Partitioning strategies for spatio-textual similarity join. In Proc of the 3rd ACM Int. Workshop on Analytics for Big Geospatial Data, pages 40-49, 2014

[6] G. Tsatsanifos and A. Vlachou. On processing top-k spatio-textual preference queries. In Proc. of the 18th EDBT Conf., pages 433-444, 2015

[7] Suphakit Niwattanakul*, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu: Using of Jaccard Coefficient for Keywords Similarity, Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong

Παραρτήματα

Παράρτημα Α - Apache Spark

Για την υλοποίηση του παραπάνω αλγορίθμου χρησιμοποιήθηκε το εργαλείο Apache Spark.

Το Spark[3] είναι ένα εργαλείο το οποίο αναπτύχθηκε το 2009 στα πλαίσια ενός ερευνητικού προγράμματος στο οποίο μετείχαν τα πανεπιστήμια της California και του Berkeley. Το εργαλείο αυτό βασίστηκε στην λογική του Hadoop Map-Reduce με στόχο να είναι πιο φιλικό προς το χρήστη. Επιπλέον, το Spark υποστηρίζει την υλοποίηση διαδραστικών ερωτημάτων στη βάση δεδομένων ή σε κάποιο dataframe και επεξεργασία δεδομένων που έχουν συνεχόμενη ροή (streaming data). Οι γλώσσες προγραμματισμού που υποστηρίζονται από αυτό είναι η Java,Scala,Python,R.

Οι βιβλιοθήκες από τις οποίες αποτελείται στο Spark υποστηρίζουν:

- Την δημιουργία εφαρμογών Μηχανικής Μάθησης (MLlib)
- Την δυνατότητα επεξεργασίας δεδομένων συνεχούς ροής(Spark Streaming)
- Την επεξεργασία γράφων(GraphX)

Εκτός από τις βιβλιοθήκες που περιέχει το Spark, ένα δεύτερο σημαντικό μέρος του είναι το Spark Core. που αποτελεί την καρδιά του Apache Spark και είναι υπεύθυνο για την παροχή των εξής λειτουργιών: την μετάδοση(transmission) εργασιών, τον προγραμματισμό(scheduling) και τις λειτουργίες εισόδου-εξόδου των δεδομένων σε ένα κατανεμημένο περιβάλλον. Ο πιο συνήθης τύπος δεδομένων για την επίτευξη αυτών των λειτουργιών είναι RDD (Resilient Distributed Dataset) που είναι σχεδιασμένα να αποκρύπτουν από το χρήστη περίπλοκους υπολογισμούς για την κατανομή των δεδομένων ώστε να επιτυγχάνεται αυτή μέσω εύκολων χειρισμών εκ μέρους του χρήστη. Ένας εύκολος χειρισμός από την πλευρά του χρήστη είναι το partitioning των δεδομένων όπου ο χρήστης μέσω ενός υπολογιστή που αποτελεί τον master σε ένα δίκτυο υπολογιστών, επιλέγει τα σημεία τομής των δεδομένων και τον τρόπο τομής, Στη συνέχεια ο master διαμοιράζει αυτά τα κομμάτια των δεδομένων στους υπόλοιπους υπολογιστές του δικτύου slaves για την περαιτέρω επεξεργασία τους.

Τα πλεονεκτήματα του Apache Spark είναι τα εξής:

- Έχει υψηλή ταχύτητα επεξεργασίας δεδομένων. Λόγω του ότι τα δεδομένα που φορτώνονται στο δίκτυο των υπολογιστών αποθηκεύονται στη ram των υπολογιστών και εφόσον εξαντλήσουν τον χώρο που παρέχει η ram προχωρούν στην αποθήκευση των δεδομένων στον σκληρό δίσκο. Αυτό το κριτήριο κάνει εξαιρετικά γρήγορο τον χρόνο προσπέλασης των δεδομένων λόγο του ότι η ram έχει πολύ υψηλή ταχύτητα προσπέλασης των δεδομένων από τον επεξεργαστή σε σχέση με τον σκληρό δίσκο, όπου πρέπει να φορτωθούν τα δεδομένα από τον σκληρό δίσκο στην ram και στην συνέχεια να προσπελαστούν από τον επεξεργαστή. Η συγκεκριμένη τεχνολογία κάνει το Spark 100 φορές πιο γρήγορο από την MapReduce όταν τα δεδομένα βρίσκονται στην ram και 10 φορές πιο γρήγορο όταν τα δεδομένα βρίσκονται στον σκληρό δίσκο στο μεγαλύτερο πλήθος πειραμάτων.

[3] https://databricks.com/glossary/what-is-apache-spark

- Επεξεργάζεται συνεχούς ροής δεδομένα. Μπορεί να επεξεργαστεί συνεχούς ροής δεδομένα (streaming data) χωρίζοντας αυτά σε μικρά κομμάτια (mini-batches) και εκτελώντας μετασχηματισμούς RDD σε αυτά.
- Σε μια εφαρμογή μπορούν να συνδυαστούν πολλαπλοί τρόποι επεξεργασίας των δεδομένων: interactive querry, real-time analytics, machine learning, graph processing, sql querries.
- Εύκολη χρηστικότητα λόγω του ότι μπορεί να υποστηριχθεί από 4 γλώσσες προγραμματισμού: Java, Scala, Python, R

Παράρτημα Β – Τεχνολογίες Διαδικτυακής Πλατφόρμας

Flask

Το Flask[4] είναι ένα web-framework της python και χρησιμοποιείται για την δημιουργία web εφαρμογών και κατατάσσεται στην κατηγορία των micro-frameworks επειδή δεν περιλαμβάνει στην βασική του έκδοση ORM(Object Relation Mappers). Όμως, μέσω της βιβλιοθήκης Sql-Alchemy παρέχεται αυτή η λειτουργεία.

Πιο συγκεκριμένα, ξεκινώντας την πιο εκτεταμένη ανάλυση του web-framework Flask. Ξεκινάμε την επεξήγηση του τι είναι web-framework. Web Framework ή Web Application Framework είναι μια συλλογή από βιβλιοθήκες και εντολές για την ενεργοποίηση προγραμματικών εφαρμογών ιστού που στοχεύον στην εύκολη συγγραφή διαδικτυακών εφαρμογών. Χωρίς να χρειάζεται ο προγραμματιστής με χαμηλού προγραμματιστικού επιπέδου λειτουργίες όπως είναι η διαχείριση νημάτων του επεξεργαστή.

Το Flask όπως αναφέρθηκε είναι ένα web application framework γραμμένο σε python που δημιουργήθηκε από τον Armin Ronacher, ο οποίος ήταν επικεφαλής μιας διεθνούς ομάδας από προγραμματιστές python και ονομαζόταν Poocco. Το Flask βασίζεται στο Werkzeug WSGI toolkit και στο Jinja 2 template engine που και τα δύο ήταν project των Poocco. WSGI είναι μια διεπαφή ανάμεσα στον web-server και στην web-application. Το Werkzeug είναι ένα WSGI toolkit το οποίο εκτελεί request, response από αντικείμενα και χρήσιμες συναρτήσεις. Αυτές τις δυνατότητες τις δίνει ένα web-framework, το οποίο έχει δημιουργηθεί πάνω στο Werkzeug. Jinja 2 είναι ένας δημοφιλής template engine για την python, όπου ένα template με μια προκαθορισμένη πηγή δεδομένων την μετατρέπει σε μια δυναμική ιστοσελίδα.

Παρακάτω παρατίθεται μια απλή εφαρμογή για την καλύτερη κατανόηση του Flask. Όπως στα περισσότερα tutorials έτσι και εδώ θα δημιουργήσουμε μια Hello World εφαρμογή. Ο κώδικας Flask είναι ο εξής:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Παρατηρούμε την δήλωση του Flask, την δημιουργία μια συνάρτησης Hello World, η οποία επιστέφει το μήνυμα Hello World και στο τέλος την εντολή τρεξίματος. Παρατηρούμε επίσης και μια εντολή app.route(/) η οποία δηλώνει το template στο οποίο απευθύνεται, εφόσον είναι συνδεδεμένο με κάποιο template που εμφανίζεται και στο URL. Για να εκτελεστεί το παραπάνω κομμάτι κώδικα θα πρέπει να

[4] https://pythonbasics.org/what-is-flask-python/

υποθηκευθεί σε ένα py αρχείο, για παράδειγμα server.py και να εκτελεστεί η εντολή python server.py όπου ξεκινά ένας τοπικός server και στη συνέχεια αν από οποιοδήποτε περιηγητή μπούμε στην διεύθυνση localhost:5000 θα εμφανιστεί το μήνυμα Hello World.

HMTL

HTML[5] σημαίνει Hypertext Markup Language. Η Html δίνει τη δυνατότητα στους χρήστες να δημιουργήσουν και να δομήσουν ενότητες, παραγράφους και επικεφαλίδες για διαδικτυακές σελίδες και εφαρμογές. Η html δεν είναι μια γλώσσα προγραμματισμού διότι δεν μπορεί να δημιουργήσει δυναμικές διεργασίες. Όμως μπορεί να δημιουργήσει και να οργανώσει format εγγράφων όπως το Microsoft Word. Για παράδειγμα μπορείς να γράψεις μια πρόταση στην html και βάζοντας τα tag protasis να την δηλώσεις σαν παράγραφο.

Ξεκινώντας την ανάλυση της Html ενδιαφέρον παρουσιάει η ιστορία και το πώς ανακαλύφθηκε. Η html εφευρέθηκε από τον Tim Berners_Lee, ο οποίος ήταν φυσικός ερευνητής του CERN στο ινστιτούτο της Ελβετίας. Αυτός συνέλαβε την ιδέα για ένα Internet-based hypertext system. Hypertext είναι ένα κείμενο που περιέχει διασυνδέσεις με άλλα κείμενα όπου οι χρήστες μπορούν να έχουν άμεση πρόσβαση. Η πρώτη έκδοση της html που κοινοποιήθηκε αποτελούταν από 18 html tags και ήταν το 1991. Από τότε κάθε νέα έκδοση εμπλουτίζεται με νέα tags. Σύμφωνα με το δίκτυο προγραμματιστών της Mozilla τώρα πλέον η html έχει 140 tags, αν και μερικά από αυτά δεν υποστηρίζονται από όλους τους περιηγητές. Βέβαια η μεγαλύτερη αναβάθμιση της html έγινε το 2014 με την html5 όπου προστέθηκαν σχηματικά tag κειμένου όπως το <article>,<header>,<footer>. Πλέον η html είναι η πιο διαδεδομένη γλώσσα προγραμματισμού για τον front-end κομμάτι ιστοσελίδων.

Μια ιστοσελίδα αποτελείται από html documents που είναι αρχεία με την κατάληξη html και μπορούν να διαβαστούν από περιηγητές όπως είναι ο Safari , ο Google Chrome και ο Mozilla Firefox. Οι περιηγητές διαβάζουν τα αρχεία και προβάλλουν στους χρήστες το περιεχόμενό τους. Συνήθως κάθε ιστοσελίδα αποτελείται από την home page, about pages, contact pages τα οποία είναι διαφορετικά html αρχεία. Επίσης, κάθε html αποτελείται από ένα σύνολο από tag που ονομάζονται elements τα οποία συμβάλουν στα χρήσιμα blocks από web pages. Αυτά δημιουργούν μια ιεραρχική δομή που περιέχει ενότητες, παραγράφους, επικεφαλίδες και άλλα περιεχόμενα του block. Τα περισσότερα html elements ανοίγουν και κλείνε με tag όπως φαίνεται στο παρακάτω παράδειγμα. Επισυνάπτεται ένα κομμάτι html κώδικα για να φανεί πως δομούνται τα html elements.

[5] https://www.hostinger.com/tutorials/what-is-html

```
<div>
  <h1>The Main Heading</h1>
  <h2>A catchy subheading</h2>
  <p>Paragraph one</p>
  <img src="/" alt="Image">
  <p>Paragraph two with a <a
href="https://example.com">hyperlink</a></p>
</div>
```

Τα περισσότερα html element διαχωρίζονται με το tag: που χρησιμοποιείται για μεγαλύτερα τμήματα από ενότητες. Επίσης παρατηρούμε ότι υπάρχει μια επικεφαλίδα <h1></h1>, μια υποκεφαλίδα <h2></h2>, μια παράγραφος και μια εικόνα <img>. Στο image tag έχει δύο γνωρίσματα: src είναι η διαδρομή της εικόνας που είναι αποθηκευμένη και alt είναι η περιγραφή της εικόνας. Στην δεύτερη παράγραφο <a> υπάρχει ένα link href για οδηγήσει στην αντίστοιχη σελίδα του url.

Flask API

Για την επικοινωνία της python με την html χρησιμοποιήθηκαν APIs. Ο όρος API[6] είναι αρκτικόλεξο της φράσης Application Programming Interface και απευθύνεται σε ένα κομμάτι κώδικα που είναι σχεδιασμένο για να επικοινωνεί μια εφαρμογή με μια άλλη, σε αντίθεση με μια διεπαφή που είναι σχεδιασμένη για να επικοινωνεί ένας χρήστης με μια εφαρμογή. Ένα πρόγραμμα του υπολογιστή συχνά χρειάζεται να επικοινωνεί με το λειτουργικό σύστημα ή με ένα άλλο πρόγραμμα και η μόνη λύση σε αυτό το πρόβλημα είναι το API.

Η δημιουργία του API κρίνεται απαραίτητη όταν:

- Οι χρήστες χρειάζονται πρόσβαση στα δεδομένα σε πραγματικό χρόνο, για παράδειγμα προβολή των δεδομένων σε έναν άλλο ιστότοπο.
- Τα δεδομένα ενημερώνονται συχνά
- Όταν οι χρήστες πρέπει να έχουν πρόσβαση στα δεδομένα ανά πάσα στιγμή
- Όταν οι χρήστες πρέπει να εκτελέσουν ενέργειες στα δεδομένα όπως είναι η ενημέρωση ή η διαγραφή των δεδομένων.

Οι πιο βασικές έννοιες κατά την χρήση των API είναι οι εξής:

- **HTTP(Hypertext Transfer Protocol)** είναι η κύρια μέθοδος επικοινωνίας των δεδομένων με το διαδίκτυο. HTTP εκτελεί έναν αριθμό από μεθόδους για το ποια δεδομένα να μεταφερθούν και τι πρέπει να συμβεί όταν μεταφερθούν. Οι κοινές μέθοδοι είναι η GET που σημαίνει πάρε τα δεδομένα από κάπου και η POST που σημαίνει βάλε τα δεδομένα κάπου.
- **URL(Uniform Resource Locator)** είναι μια διεύθυνση για μια πηγή από δεδομένα στο διαδίκτυο. Ένα URL αποτελείται το protocol(https://www.unipi.gr/unipi/el/), το domain(unipi.gr) και ένα

[6] https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask#what-is-an-api

προαιρετικό path(unipi/el). Ένα url περιγράφει μια τοποθεσία από συγκεκριμένες πηγές για παράδειγμα μια ιστοσελίδα.

- **JSON(Javascript Object Notation)** είναι μια μορφή text-based data store που είναι σχεδιασμένη να διαβάζεται και από τον άνθρωπο και από την μηχανή. JSON είναι το πιο κοινό format για να επιστραφούν δεδομένα μέσα από ένα API.
- **REST(Representation State Transfer)** είναι μια μεθοδολογία που περιγράφει μερικές από τις καλύτερες πρακτικές APIs. Ένα API επιτρέπει την αμφίδρομη μεταφορά των δεδομένων.

Plotly

Plotly[7] είναι μια βιβλιοθήκη της python που περιέχει πολλών ειδών plots και τα προβάλλει σε μια html σελίδα. Στην συγκεκριμένη Διπλωματική έγινε χρήση της plotly για να προβληθούν σημεία του dataset πάνω σε χάρτη. Παρακάτω παρατίθεται μερικές πιο γενικές πληροφορίες για την Plotly.

Η Plotly αποτελείται από τρία διαφορετικά Python Apis

- **Object Oriented API** το οποίο μοιάζει με την matplotlib της python.
- **Data Driven API** που βασίζεται στην δημιουργία plot από αρχεία JSON
- **Plotly Express API** που περιέχει πιο υψηλού επιπέδου συναρτήσεις για plot όπως η Seaborn της python.

Για τις ανάγκες της πλατφόρμας που δημιουργήθηκε χρησιμοποιήθηκαν η Object Oriented API και η Plotly Express API.

PySpark

Το Apache Spark έχει δημιουργηθεί σε Scala γλώσσα προγραμματισμού. Το Pyspark[8] δημιουργήθηκε για να μπορεί να υποστηρίζει σε περιβάλλον Apache Spark να δημιουργούνται Python εφαρμογές. Δηλαδή το Pyspark είναι ένα API που υποστηρίζει τη συγγραφή python σε περιβάλλον Apache Spark. Επιπλέον το Pyspark επιτρέπει την διασύνδεση των Resilient Distributed Datasets(RDDs) στο Apache Spark με την γλώσσα προγραμματισμού Python. Αυτό γίνεται μέσω της βιβλιοθήκης Py4j.

Μερικές από τις πιο διαδεδομένες βιβλιοθήκες του Pyspark είναι:

- **PysparkSQL** είναι μια βιβλιοθήκη του Pyspark που επιτρέπει τη συγγραφή queries όπως στην SQL για να προσπελάσουν είτε δομημένα είτε ημιδομημένα δεδομένα. PysparkSQL είναι ένας wrapper πάνω από ένα Pyspark πυρήνα. Επίσης το PysparkSQL επιτρέπει στα dataframes μια πινακοειδής αναπαράσταση πάνω από δομημένα δεδομένα.
- **MLlib** είναι ένας wrapper πάνω στο Pyspark και αυτό περιέχει τις βιβλιοθήκες για machine learning σε Spark. Αυτή η βιβλιοθήκη χρησιμοποιεί τεχνικές για την παραλληλοποίηση των δεδομένων τόσο στο κομμάτι της επεξεργασίας τους όσο και της αποθήκευσης. Η MLlib

[7] https://opensource.com/article/20/5/plotly-python

[8] https://databricks.com/glossary/pyspark

υποστηρίζει πολλούς machine learning αλγορίθμους για: classification, regression, clustering, collaborative filtering, dimensionality reduction και άλλα βασικά πρωτόκολλα βελτιστοποίησης.

- **GraphFrames** έχουν σκοπό την επεξεργασία γράφων που παρέχεται μέσα από ένα σύνολο από APIs για την εκτέλεση αποδοτικής ανάλυσης γράφων χρησιμοποιώντας το Pyspark core και το PysparkSQL