Ατομική Διπλωματική Εργασία

ΤΕΧΝΙΚΕΣ ΥΠΟΛΟΓΙΣΜΟΥ ΣΗΜΑΣΙΟΛΟΓΙΚΑ ΚΟΝΤΙΝΩΝ ΧΡΗΣΤΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ ΚΟΙΝΩΝΙΚΗΣ ΔΙΚΤΥΩΣΗΣ

Ραφαήλ Κωνσταντίνου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΙΚΕΣ ΥΠΟΛΟΓΙΣΜΟΥ ΣΗΜΑΣΙΟΛΟΓΙΚΑ ΚΟΝΤΙΝΩΝ ΧΡΗΣΤΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ ΚΟΙΝΩΝΙΚΗΣ ΔΙΚΤΥΩΣΗΣ

Ραφαήλ Κωνσταντίνου

Επιβλέπων Καθηγητής
Δημήτρης Ζεϊναλιπούρ

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2014

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον κ. Ζειναλιπούρ Δημήτρη, επιβλέπον καθηγητή της παρούσας διπλωματικής εργασίας, για την υποστήριξη και καθοδήγηση που μου παρείχε κατά την διάρκεια της εκπόνησης της διπλωματικής μου εργασίας. Θερμές ευχαριστίες θα ήθελα επίσης να δώσω και στον υποψήφιο διδάκτορα Κωνσταντίνο Κώστα του οποίου η συνεισφορά του θεωρείται ανεκτίμητη. Πρόκειται για δύο σπουδαίους ανθρώπους με πολλές γνώσεις σε όλους του τομείς της Επιστήμης της Πληροφορικής.

Επίσης θα ήθελα να πω ένα μεγάλο ευχαριστώ στην οικογένεια μου και ιδιαίτερα στους γονείς μου Νίκο και Ανδρούλλα όπου με πολύ κόπο και μόχθο με έχουν μεγαλώσει και μου έχουν δώσει αρχές και γνώσεις που θα με βοηθήσουν μετέπειτα στη ζωή μου. Ένα μεγάλο ευχαριστώ και στους φίλους μου εντός και εκτός Πανεπιστημίου οι οποίοι μου συμπαραστάθηκαν όλο αυτό τον καιρό και που με συμβούλευαν σχετικά με το άγχος το οποίο με κατέβαλε κατά καιρούς.

Θα ήθελα να αφιερώσω αυτή τη διπλωματική εργασία στο πατέρα μου Νίκο (48), το θείο Κυριάκο (58), τη θεία Μαρία (52) και στη ξαδέρφη Κωνσταντίνα (21) οι οποίοι έχουν χάσει τη μάχη για ζωή από την επάρατη νόσο και οι οποίοι ήταν δίπλα μου σε κάθε μου δυσκολία.

Τέλος οφείλω να πω ένα μεγάλο ευχαριστώ στους τραγουδιστές Μιλτιάδη Πασχαλίδη, Σωκράτη Μάλαμα, Χρίστο Θηβαίο, Θανάση Παπακωνσταντίνου, Πύξ Λάξ, και πολλούς άλλους έντεχνους καλλιτέχνες. Ακούγοντας τα τραγούδια τους ερχόταν η ηρεμία όπου έφερνε την αυτοσυγκέντρωση, συστατικά που με έχουν βοηθήσει να αντεπεξέλθω των δυσκολιών που συνάντησα στα τέσσερα χρόνια των σπουδών μου.

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη και η παρουσίαση τεχνικών για υπολογισμό των σημασιολογικά κοντινότερων χρηστών σε συστήματα κοινωνικής δικτύωσης. Συγκεκριμένα θα προταθεί ένας αλγόριθμος για την πλατφόρμα κοινωνικής δικτύωσης Rayzit, ο οποίος θα προβαίνει σε συστάσεις, για τους σημασιολογικά κοντινούς χρήστες οι οποίοι είναι ενεργοί στη πλατφόρμα. Ο αλγόριθμος αξιοποιεί τα ενδιαφέροντα του κάθε χρήστη όπως αυτά έχουν εκφραστεί μέσα από τα μηνύματα τα οποία έχει κατά καιρούς ανταλλάξει στην πλατφόρμα. Λαμβάνοντας υπόψη το ιστορικό του χρήστη, έχει αναπτυχθεί ένα σύστημα συστάσεων το οποίο θα αξιολογήσει τις συστάσεις όχι μόνο σε σχέση με τα ενδιαφέροντα των χρηστών αλλά και με την πραγματική απόσταση που έχουν οι χρήστες μεταξύ τους.

Για την αποπεράτωση της διπλωματικής εργασίας έχει μελετηθεί βιβλιογραφία, που δείχνει την αναγκαιότητα ύπαρξης συστημάτων σύστασης σε συστήματα κοινωνικών δικτύων αλλά και σε συστήματα αγοραπωλησίας προϊόντων. Επιπλέον έχουν αξιολογηθεί τα προβλήματα που υπάρχουν σε αυτά τα συστήματα με σκοπό την αντιμετώπιση τους και στο Rayzit.

Το σύστημα συστάσεων που έχει υλοποιηθεί απαριθμεί δύο προσεγγίσεις, όπου στη πρώτη χρησιμοποιείται η δομή δεδομένων Inverted Index και στη δεύτερη μια κατά προσέγγιση επίλυση που κάνει χρήση MinHash υπογραφών. Οι δύο αυτές προσεγγίσεις αξιολογούνται με βάση το συνολικό χρόνο εκτέλεσης συναρτήσει του αριθμού των χρηστών που λαμβάνουν μέρος στα πειράματα αλλά και με παράγοντες που καθορίζουν την ορθότητα που είναι το false positives στη περίπτωση της χρήσης MinHash υπογραφών. Γίνεται ανάλυση της πολυπλοκότητας των δύο προσεγγίσεων, η οποία επιβεβαιώνεται μέσα από τη πειραματική αποτίμηση.

Ακολούθως θα γίνει παρουσίαση μιας εφαρμογής που αναπτύχθηκε στη πλατφόρμα Android για να υποστηρίξει το Rayzit για χρήστες που χρησιμοποιούν το Android OS αλλά και για να γίνει πραγματική αξιολόγηση του συστήματος συστάσεων μέσα από τη χρήση της εφαρμογής.

Τέλος, παρουσιάζονται συμπεράσματα και μελλοντικές επεκτάσεις οι οποίες δύναται να δώσουν καλύτερα αποτελέσματα στο μέλλον όσων αφορά τις συστάσεις. Επίσης, δίνονται λύσεις όπου χρησιμοποιώντας τα υφιστάμενα αποτελέσματα μπορεί να γίνει επέκταση του συστήματος συστάσεων .

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Σε αυτό το κεφάλαιο θα επεξηγηθούν οι βασικοί λόγοι που προβλημάτισαν και παρακίνησαν τη μελέτη του συγκεκριμένου θέματος που θα παρουσιαστεί στη συνέχεια. Θα δοθεί η παρακίνηση του προβλήματος αλλά και ποιο θα είναι το τελικό αποτέλεσμα της εν λόγω μελέτης. Αρχικά θα δοθούν κάποιες επεξηγήσεις που αφορούν τη πλατφόρμα Rayzit και πως αυτή στη συνέχεια θα συσχετιστεί με το σκοπό της διπλωματικής εργασίας. Σε επόμενη υποενότητα θα αναφερθούν και θα επεξηγηθούν με σαφήνεια οι προκλήσεις που κλήθηκε να αντιμετωπίσει η πλατφόρμα Rayzit και τις λύσεις που έχουν προταθεί για τα συγκεκριμένα προβλήματα. Εν κατακλείδι, θα δοθεί ένα γενικό περίγραμμα της Ατομικής Διπλωματικής Εργασίας το οποίο θα δίνει στον αναγνώστη την πορεία και τη μεθοδολογία που ακολουθήθηκε για την αντιμετώπιση και τη λύση του προβλήματος που λύνει η συγκεκριμένη εργασία.

1.1 Προοίμιο

Το πρόβλημα της υλοποίησης και της σωστής λειτουργίας ενός συστήματος συστάσεων, που είναι γνωστό με την αγγλική ορολογία ως «recommendation system», αποτελεί αναπόσπαστο κομμάτι, για όλες τις σύγχρονες εταιρείες Πληροφορικής οι οποίες ειδικεύονται σε παροχή υπηρεσιών προς τους χρήστες τους.

Οι υπηρεσίες αυτές μπορεί να είναι διαφόρου τύπου. Για παράδειγμα υπηρεσίες που παρέχονται στις μέρες μας, και βρίσκουν αξιόλογο ενδιαφέρον από το κοινό που τις χρησιμοποιεί είναι τα κοινωνικά δίκτυα. Επίσης, μία άλλη υπηρεσία που παρέχεται και είναι ευρέως διαδεδομένη, είναι οι υπηρεσίες συναλλαγών που έχουν στο επίκεντρο τους τις αγοραπωλησίες των χρηστών.

Διάφορες εταιρείες όπως το Twitter και το Facebook που είναι κατεξοχήν εταιρείες που ειδικεύονται σε παροχή υπηρεσιών κοινωνικής δικτύωσης, αλλά και η εταιρεία Amazon που είναι μια από τις ποιο δημοφιλής εταιρείες που παρέχουν υπηρεσίες αγοραπωλησιών, όλες τους χρησιμοποιούν συστήματα συστάσεων (δείτε Σχήμα 1).

Το να έχει μια εταιρεία στη διάθεση της ένα πλεονέκτημα που ουσιαστικά μπορεί να προβλέπει με σχετική επιτυχία τις προτιμήσεις που δύναται να έχει ένας χρήστης στο μέλλον, αυτό της προσδίδει ένα πολύ δυνατό εργαλείο γιατί έχει την δυνατότητα να προβλέπει τα ενδιαφέροντα του κάθε χρήστη που χρησιμοποιεί την υπηρεσία της. Αυτό της δίνει το δικαίωμα να μπορεί να προλαμβάνει τον χρήστη της από άσκοπες αναζητήσεις, που ισοδυναμεί σε άσκοπο χρόνο, έτσι αυτό τον τρόπο ο χρήστης θα μένει ικανοποιημένος από την χρήση της υπηρεσίας. Αυτό βραχυπρόθεσμα θα έχει πολύ θετικά στοιχεία για την ανέλιξη της υπηρεσίας. Περισσότερες λεπτομέρειες πάνω σε αυτό το θέμα θα αναφερθούν στο επόμενο κεφάλαιο όπου θα γίνει εκτενής αναφορά για το πώς λειτουργούν αυτά τα συστήματα.

1.2 Παρακίνηση Προβλήματος

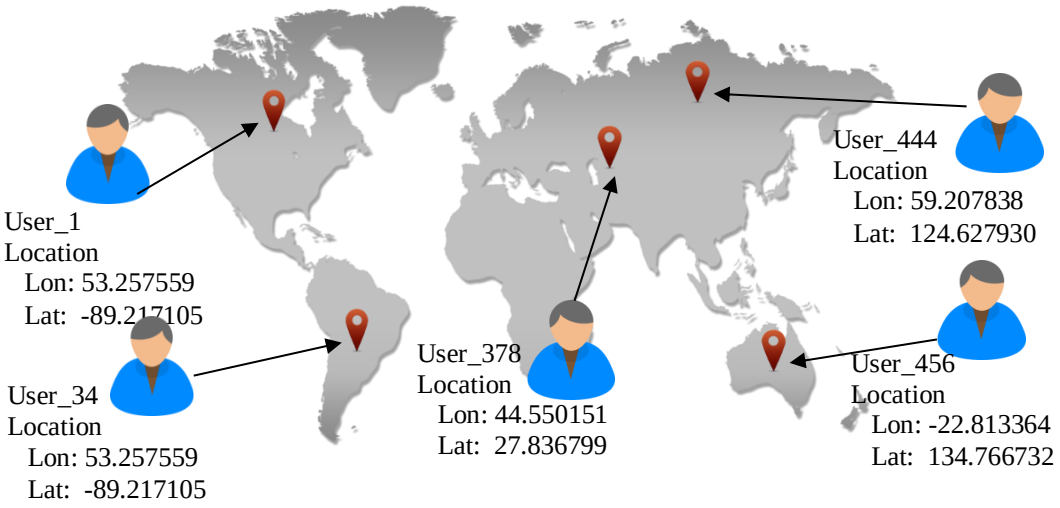
Μια υπηρεσία κοινωνικής δικτύωσης ορίζεται ως ένα δίκτυο με κοινωνικές αλληλεπιδράσεις μεταξύ των χρηστών που το χρησιμοποιούν και που αποσκοπούν σε προσωπικές σχέσεις μεταξύ τους. Ένας άλλος ορισμός ορίζει ότι είναι μια υπηρεσία που έχει σκοπό να προάγει την επικοινωνία των χρηστών μεταξύ τους, με τη χρήση των διαφόρων σχόλιων που κάνει ο ένας στον άλλον, μέσα που μηνύματα που ανταλλάζουν είτε και μέσα από εικόνες που

Σχήμα 1.1 : Παράδειγμα Συστάσεων για αγορά βιβλίων στη πλατφόρμα Amazon

βάζουν μεταξύ τους οι χρήστες. Οι σύγχρονες απαιτήσεις και ανάγκες συνεχώς αλλάζουν τον ορισμό για το τι ορίζεται μια υπηρεσία κοινωνικής δικτύωσης. Το σημαντικό που πρέπει να έχει κάποιος κατά νου είναι το ότι όλες αυτές οι υπηρεσίες χρησιμοποιούν συστήματα συστάσεων για να παρέχουν καλύτερη ποιότητα υπηρεσίας στους χρήστες τους ωθώντας έτσι όλο και περισσότερους υφιστάμενους χρήστες να συνεχίσουν να τη χρησιμοποιούν αλλά και να φέρει κοντά νέους χρήστες και να τους κρατήσει ενεργούς.

Οι ομοιότητες που έχουν οι υπηρεσίες Facebook, Twitter με την υπηρεσία Rayzit, είναι στο ότι είναι και οι τρεις τους υπηρεσίες κοινωνικής δικτύωσης. Έχουν δηλαδή στο επίκεντρο τους το χρήστη και πως μπορούν να κερδίζουν ολοένα και περισσότερους στο ενεργητικό τους ανά τακτά χρονικά διαστήματα. Η υπηρεσία Amazon, που μαζί με το Twitter και το Facebook θα μελετηθούν στο επόμενο κεφάλαιο, χρησιμοποιούν συστήματα συστάσεων για να επιτελέσουν τις διάφορες επιλογές που προσφέρουν στους χρήστες τους ως υπηρεσίες.

Αισίως καταλήγουμε στην παρακίνηση του προβλήματος που είναι αυτό που προαναφέρθηκε έμμεσα μέχρι τώρα. Η ανάγκη για καλυτέρευση της ποιότητας υπηρεσίας που παρέχεται μέχρι στιγμής στη πλατφόρμα «Rayzit», εισάγοντας καινούργιες έννοιες – πτυχές μέσα στη πλατφόρμα με τρόπο που να ωθήσει περισσότερους χρήστες να χρησιμοποιήσουν τη πλατφόρμα, αποσκοπώντας έτσι στη καλυτέρευση της ποιότητας υπηρεσίας. Η καινούργια έννοια είναι η ανάπτυξη ενός καινοτόμου συστήματος συστάσεων που θα παρέχει στους χρήστες τη δυνατότητα να τους φέρνει κοντά σε άλλους χρήστες με κοινά ενδιαφέροντα λαμβάνοντας υπόψη τη πραγματική τους απόσταση. Με αυτό τον τρόπο οι χρήστες αφού θα έρχονται ποιο κοντά θα προάγεται η κοινωνικότητα μεταξύ τους, η ανταλλαγή απόψεων και θα προσελκύσει περισσότερο κόσμο να χρησιμοποιήσει την υπηρεσία. Θέλοντας να αξιολογηθούν κάποια παραδείγματα από υφιστάμενες υπηρεσίες που ήδη χρησιμοποιούν συστήματα συστάσεων έχει γίνει η επιλογή της μελέτης των υπηρεσιών Twitter, Facebook και Amazon οι οποίες θα αξιολογηθούν στο επόμενο κεφάλαιο.



Σχήμα 1.2 : Εικονική Παρουσίαση αναπαράστασης χρηστών μέσα στο πραγματικό κόσμο

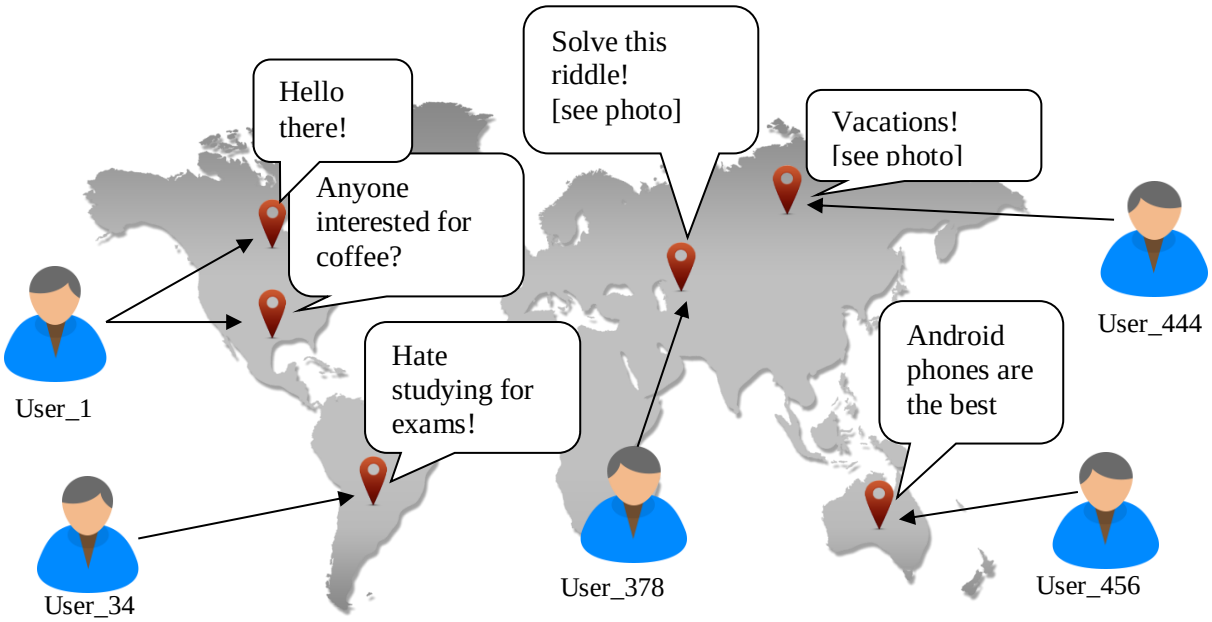
1.3 Επεξήγηση της Πλατφόρμας «Rayzit»

Πριν αναφερθούν οι διάφορες λεπτομέρειες θα πρέπει να δοθούν επαρκής εξηγήσεις οι οποίες θα πρέπει να οριοθετούν κατάλληλα το θέμα γύρω από το οποίο θα κινηθεί η συγκεκριμένη διπλωματική εργασία και να δίνουν εξηγήσεις για το πώς λειτουργεί μέχρι τώρα η υπηρεσία Rayzit και πώς αναμένεται να λειτουργεί στη συνέχεια μετά και την εφαρμογή του συστήματος συστάσεων.

Συνοπτικά όπως έχει αναφερθεί και μέχρι τώρα, είναι μια πλατφόρμα κοινωνικής δικτύωσης. Σε αυτήν λαμβάνουν μέρος διάφοροι χρήστες οι οποίοι κάθε φορά που θα ενεργοποιήσουν την εφαρμογή τους στο κινητό τους τηλέφωνο, η εφαρμογή το πρώτο πράγμα που θα κάνει είναι να ενημερώσει την υποδομή πάνω στην οποία τρέχει η υπηρεσία, για τη τελευταία γνωστή γεωγραφική θέση στην οποία βρίσκεται ο χρήστης.

Όπως παρατηρείται και στο Σχήμα 1.2 ο κάθε χρήστης έχει καταχωρημένη τη τρέχουσα τοποθεσία του μέσα στο σύστημα. Οι χρήστες σύμφωνα με το Σχήμα 1.3 καθώς γεω-τοποθετούνται μέσα στο χώρο μπορούν να ανταλλάζουν μηνύματα μεταξύ τους. Ο κάθε χρήστης μπορεί να στέλνει το μήνυμα, ερωτήσεις , θέμα προς συζήτηση, ιδέες, στους κοντινότερους χρήστες γύρω του. Με αυτό τον τρόπο παρέχεται στους χρήστες η ευκαιρία να επικοινωνήσουν μεταξύ τους και να δουν σε πραγματικό χρόνο ποιοι γύρω τους χρησιμοποιούν την υπηρεσία και ποια θέματα τους απασχολούν για συζήτηση.

Στο Σχήμα 1.3 φαίνεται η επικοινωνία και πως αυτή συντελείται στην υπηρεσία «Rayzit». Αξιοσημείωτο να τονιστεί ότι όσοι χρησιμοποιούν τη συγκεκριμένη επικοινωνία διατηρούν την ανωνυμία τους τόσο σε θέμα προσωπικών δεδομένων όσο και σε θέμα γεω-τοποθέτησης



Σχήμα 1.3 : Εικονική Παρουσίαση αναπαράστασης μηνυμάτων χρηστών μέσα στο πραγματικό κόσμο

μέσα στο πραγματικό κόσμο. Δεν μπορεί κανένας χρήστης με κανένα τρόπο να μάθει κάποιο στοιχείο για κάποιο άλλο χρήστη και ούτε ο χρήστης είναι υπόχρεος να υποβάλει στην υπηρεσία κάποια στοιχεία. Στο Σχήμα 1.3 φαίνεται μια εικονική παρουσίαση κάποιων μηνυμάτων των χρηστών μέσα στο πραγματικό κόσμο. Αξιοσημείωτο να αναφερθεί ότι οι χρήστες εκτός από μηνύματα μπορούν να ανταλλάζουν και πολυμέσα όπου με αυτό τον τρόπο η υπηρεσία παίρνει ένα ποιο διαδραστικό χαρακτήρα.

Καθώς οι χρήστες χρησιμοποιούν την υπηρεσία ολοένα και περισσότερα μηνύματα υποβάλλονται μέσα στην υπηρεσία. Το κάθε μήνυμα όταν θα ληφθεί στους εξυπηρετητές που υποστηρίζουν την υπηρεσία, θα τρέξει ο αλγόριθμος που βρίσκει τους κοντινότερους χρήστες γύρω από το χρήστη που έστειλε το μήνυμα και ακολούθως θα τους στείλει το μήνυμα στην εφαρμογή τους. Το ότι το κάθε μήνυμα που υποβάλλεται μέσα στην υπηρεσία θα σταλεί στους κοντινότερους χρήστες σε σχέση με το χρήστη που το έστειλε αυτό είναι κάτι καινοτόμο για μια υπηρεσία. Με αυτή τη προσέγγιση ουσιαστικά υιοθετούνται αλγόριθμοι εγγύτητας οι οποίοι προσδίδουν στην υπηρεσία μοναδικό χαρακτήρα.

Οι αλγόριθμοι εγγύτητας στην αγγλική βιβλιογραφία ονομάζονται ως k- nearest - problem (kNN). Μια παραλλαγή του συγκεκριμένου προβλήματος είναι το AkNN, που στην αγγλική βιβλιογραφία ορίζεται ως All k nearest neighbors. Περισσότερες λεπτομέρειες για το πρόβλημα του kNN αλλά και για το AkNN θα ακολουθήσει σε επόμενο κεφάλαιο με σχηματικά παραδείγματα.

1.4. Συνεισφορές

Μέσω της Ατομικής Διπλωματικής Εργασίας έχουν γίνει δύο σημαντικές συνεισφορές:

1. Έχει προταθεί ο αλγόριθμος για τους k Σημασιολογικά Κοντινότερους Γείτονες (k Semantic Nearest Neighbors - kSNN) ο οποίος χρησιμοποιεί Multi-Objective τεχνικές για τον υπολογισμό των γεωγραφικά κοντινότερων γειτόνων, για κάποιο χρήστη, σύμφωνα με τα ενδιαφέρονται που έχουν κατά καιρούς δείξει που χρησιμοποιούν την υπηρεσία Rayzit, μέσα στα μηνύματα τους. Ο αλγόριθμος ονομάζεται SNA (Semantic Neighbor Algorithm)

2. Υλοποιήθηκε η εφαρμογή Rayzit για τη πλατφόρμα Android OS στην οποία θα εφαρμοστεί ο SNA. Με αυτή την εφαρμογή θα παρέχεται ένα ολοκληρωμένο περιβάλλον μέσα από το οποίο ο κάθε χρήστης θα μπορεί να βλέπει και να αξιολογεί τη λειτουργικότητα του αλγορίθμου. Ο χρήστης θα μπορεί να επιλέγει για το αν θα δίνεται περισσότερη έμφαση στην γεωγραφική απόσταση ή τα ενδιαφέροντα του.

1.5 Περίγραμμα Ατομικής Διπλωματικής Εργασίας

Στο πρώτο κεφάλαιο παρουσιάστηκε το βασικό υπόβαθρο για τη συγκεκριμένη Ατομική Διπλωματική Εργασία. Συγκεκριμένα έχει παρουσιαστεί και επεξηγηθεί η παρακίνηση που ώθησε τους λόγους για τη λύση του συγκεκριμένου προβλήματος. Το πρόβλημα όπως έγινε αντιληπτό εντοπίζεται γύρω από τη πλατφόρμα Rayzit και πως μπορεί να γίνει αύξηση στη ποιότητας υπηρεσίας που ήδη παρέχεται μέχρι στιγμής στους χρήστες της. Σε αυτό το κεφάλαιο μέχρι τώρα έχει δοθεί ένα βασικό πλαίσιο εργασίας για τη πλατφόρμα Rayzit που επεξηγείται ο τρόπος λειτουργίας της και διάφορες άλλες πτυχές οι οποίες θα φανούν χρήσιμες στη πορεία. Στο δεύτερο κεφάλαιο που θα ακολουθήσει, θα παρουσιαστεί η έρευνα που έχει γίνει σχετικά με συστήματα συστάσεων και πως αυτά δύναται να βοηθήσουν στη παρόμοιου συστήματος που θα υποστηρίζει την υπηρεσία Rayzit. Ακολούθως θα παρουσιαστούν κάποια βασικά χαρακτηριστικά αλγορίθμων εγγύτητας τα οποία θα χρησιμοποιηθούν ως βάση προς μελέτη και στον αλγόριθμο που θα προταθεί στη συγκεκριμένη διπλωματική. Στο τρίτο κεφάλαιο θα παρουσιαστεί ολοκληρωμένα ο καθορισμός προβλήματος και το μοντέλο συστήματος για να μπορεί αισίως στο τέταρτο κεφάλαιο να γίνει μια εκτενής περιγραφή των προσεγγίσεων που οδήγησαν στη σύσταση του SNA αλγόριθμου. Στο πέμπτο κεφάλαιο θα παρουσιαστεί συνοπτικά η εφαρμογή που έχει γίνει για να υποστηρίξει την υπηρεσία Rayzit για τις κινητές συσκευές Android. Στη συνέχεια στο έκτο και έβδομο κεφάλαιο θα παρουσιαστεί η πειραματική μεθοδολογία και αποτίμηση των πειραμάτων που καταδεικνύουν τον ορθότητα, επεκτασιμότητα και απόδοση των δύο προσεγγίσεων του αλγορίθμου και στο τελευταίο και όγδοοσ κεφάλαιο θα παρουσιαστούν τα συμπεράσματα και μελλοντικές προτάσεις που αφορούν το συγκεκριμένο πρόβλημα.

Κεφάλαιο 2

Πλατφόρμες Κοινωνικής Δικτύωσης

Στο κεφάλαιο αυτό θα αναφερθούν όλοι οι τεχνικοί όροι που αφορούν το αντικείμενο των κοινωνικών δικτύων και ποια στοιχεία τα συνθέτουν. Ακολούθως θα γίνει μια αναφορά για το πώς οι σύγχρονες πλατφόρμες μοιάζουν σε σχέση με τη πλατφόρμα Rayzit. Στη συνέχεια θα παρουσιαστούν συνοπτικά κάποια στοιχεία που συνθέτουν τα συστήματα που αφορούν αγοραπωλησίες και πως αυτά βρίσκουν ομοιότητα με το πρόβλημα που πρέπει να επιλυθεί.

2.1 Πλατφόρμες Κοινωνικής Δικτύωσης

Η πλατφόρμα Rayzit, όπως το Facebook, Twitter και πολλές άλλες, είναι πλατφόρμες κοινωνικής δικτύωσης. Ο πλήρης ορισμός για το τι είναι μια πλατφόρμα κοινωνικής δικτύωσης έχει δοθεί στο προηγούμενο κεφάλαιο για να δώσει ένα βασικό υπόβαθρο. Σε αυτό το κεφάλαιο και συγκεκριμένα σε αυτή την υποενότητα θα δοθούν κάποια ποιο επεξηγηματικά χαρακτηριστικά και μια πλήρης εικόνα για τις πλατφόρμες κοινωνικής δικτύωσης και πως αυτές υλοποιούν συστήματα συστάσεων.

Η έρευνα έχει εστιαστεί στο Facebook και στο Twitter. Έχουν επιλεγεί οι συγκεκριμένες που προαναφέρθηκαν για το λόγο ότι είναι αρκετά δημοφιλείς στο κοινό και θα μας δώσουν ίσως ένα καλύτερο μέτρο σύγκρισης αφού συγκεντρώνουν αρκετά στοιχεία που τις κάνουν μοναδικές. Το επίκεντρο σε αυτές τις υπηρεσίες είναι οι χρήστες. Οι χρήστες συχνά στις μέρες μας αποζητούν οι υπηρεσίες που χρησιμοποιούν να έχουν προφανώς κάποια βασικά χαρακτηριστικά που θα πρέπει να διέπουν κάθε υπηρεσία κοινωνικής δικτύωσης, αλλά επίσης απαιτούν αυτές οι υπηρεσίες, να έχουν και κάποια χαρακτηριστικά που τις κάνουν μοναδικές στο είδος τους.

Η μοναδικότητα ως προς το είδος υπηρεσίας που προσφέρει μια πλατφόρμα κοινωνικής δικτύωσης σε σχέση με κάποια άλλη είναι το μέτρο σύγκρισης που καθορίζει το πόσο αρεστή είναι μια υπηρεσία. Το Rayzit μέχρι τώρα κάνει χρήση του kNN και αναμένεται να κερδίσει επιπρόσθετη μοναδικότητα χρησιμοποιώντας ένα σύστημα συστάσεων.

Για παράδειγμα μπορούμε να συγκρίνουμε το Twitter, το Facebook μαζί με το Rayzit. Οι διαφορές τους είναι αρκετές. Κάποιες διαφορές όμως που είναι αρκετά αξιόλογες και κάνουν τη διαφορά είναι οι εξής και φαίνονται στο Πίνακα 2.1.

Facebook , Twitter and Rayzit Features

	Facebook	Twitter	Rayzit
Features	Facebook features include the, Friends, Fans, Wall, News Feed, Fan Pages, Groups, Apps, Live Chat, Likes, Photos, Videos, Text, Polls, Links, Status, Pokes, Gifts, Games, Messaging, Classified section, upload and download options and others	Tweet, Retweet, Direct Messaging, Follow People & Trending Topics, Links, Photos, Videos	Send "Rayz" messages (text, audio, video or image) to the crowd around you (typically a few hundred people that are closest to you). "Re-Rayz" a message you receive to spread the word about important things. "Star" a message to follow a conversation wherever you go.
Introduction	Facebook is a social networking service launched in February 2004, owned and operated by Facebook, Inc. As of September 2013, Facebook has over one billion active users, more than half of whom use Facebook on a mobile device.	Twitter is an online social networking service and microblogging service that enables its users to send and read text-based messages of up to 140 characters, known as "tweets".	Rayzit is an award-winning crowd messaging technology that delivers your questions, inquiries and ideas to the closest users around, regardless of how far these users are and who they are.
Advertising	Advertising supported in the form of banner ads, referral marketing, casual games	Advertising supported in the form of promoted tweets	NO
Instant messaging	**YES**	**NO**	**YES**
Play games	YES	NO	NO
Users express approval of content by	"Like"	"Retweet" or "Favorite"	"Rerayz" or "Star"
Post length	Unlimited	Limited	Limited

Πίνακας 2.1 : Διαφορές μεταξύ των υπηρεσιών κοινωνικής δικτύωσης Facebook,Twitter και Rayzit

Αν μελετηθεί ο Πίνακας 2.1 μπορούμε να παρατηρήσουμε ότι οι διαφορές μεταξύ των τριών υπηρεσιών κοινωνικής δικτύωσης είναι αρκετές. Είναι όμως αυτές οι διαφορές που τις κάνουν να είναι αρεστές και προσιτές στο συγκεκριμένο κόσμο που τις προτιμά (target group). Αρκεί μόνο να προσέξουμε πόσες περισσότερες επιλογές παρέχει το Facebook στο κοινό του σε σχέση με το Twitter και όμως το Twitter απαριθμεί περισσότερους ενεργούς χρήστες απ' ότι το Facebook.

2.2 Πλατφόρμες Αγοραπωλησίας

Μια πλατφόρμα αγοραπωλησίας ορίζεται ως μια πλατφόρμα υπηρεσιών που ειδικεύεται στη πώληση προϊόντων. Οι διάφοροι χρήστες που χρησιμοποιούν την υπηρεσία μπορούν είτε να αγοράζουν είτε να πωλούν διάφορα προϊόντα. Υπάρχουν πάρα πολλά παραδείγματα τέτοιων υπηρεσιών. Τα ποιο αντιπροσωπευτικά είναι αυτά του Amazon και του Ebay. Τα συστήματα αυτά τον τελευταίο καιρό έχουν δείξει πραγματικά σημάδια ανέλιξης και πολλοί χρήστες τα προτιμούν λόγω της ποιότητας υπηρεσιών που τους παρέχουν.

Και οι δύο εταιρείες χρησιμοποιούν συγκεκριμένα συστήματα σύστασης ,τα οποία επειδή οι υπηρεσίες που προσφέρουν μοιάζουν, ο τρόπος συμπεριφοράς των συστημάτων σύστασης τόσο του Amazon όσο και του Ebay μοιάζουν κινούνται στο ίδιο πλαίσιο.

2.3 Συστήματα Συστάσεων και τρόπος Λειτουργίας

Στη συγκεκριμένη υποενότητα θα παρουσιαστούν τα συστήματα σύστασης που χρησιμοποιούνται από το Twitter τα οποία σε αρκετό βαθμό μοιάζουν με αυτά του Facebook. Γι' αυτό θα αναλυθούν αποκλειστικά μόνο του Twitter, που σαν υπηρεσία έχει περισσότερες ομοιότητες με το Rayzit σε σχέση με το Facebook, όπως αυτό συμπεραίνεται εξετάζοντας τον Πίνακα 2.1 . Στην συνέχεια θα παρουσιαστεί αντίστοιχα πως το Amazon χρησιμοποιεί ένα τέτοιο σύστημα το οποίο όπως αναφέρθηκε έχει αρκετές ομοιότητες με το Ebay.

2.3.1 Twitter και Συστήματα Συστάσεων

Στο Twitter ανεξάρτητα με άλλα κοινωνικά δίκτυα οι σχέσεις μεταξύ των χρηστών μπορεί να είναι κοινωνικού, επικοινωνιακού, και πληροφοριακού τύπου ή και συνδυασμός τους. Αυτό συμβαίνει γιατί οι χρήστες ακολουθούν (follow) ο ένας τον άλλο και με αυτό τον τρόπο διατηρούνται κοινωνικοί σύνδεσμοι μεταξύ των χρηστών (social links) και μπορούν μεταξύ τους να ανταλλάζουν χρήσιμες πληροφορίες.

Ταξινομία	Περιγραφή
Tweet	Το Tweet αναφέρεται στο τι περιέχει το κείμενο (#HashTags , URL:links)
Retweet	Το Retweet συμβαίνει όταν ένας χρήστης προωθεί σε αυτούς που τον ακολουθούν ένα μήνυμα από άλλους χρήστες
Follow	Το Follow συμβαίνει όταν κάποιος θέλει να ακολουθεί κάποιο άλλο χρήστη (δημιουργεί σύνδεσμο) και να λαμβάνει ενημερώσεις από αυτόν. Ο χρήστης που δημιουργεί το σύνδεσμο ονομάζεται follower και ο άλλος followee.
Mention	Τρόπος που ο χρήστης τυγχάνει προσοχής από ένα άλλο χρήστη.

Πίνακας 2.1 : Ταξινομία Twitter

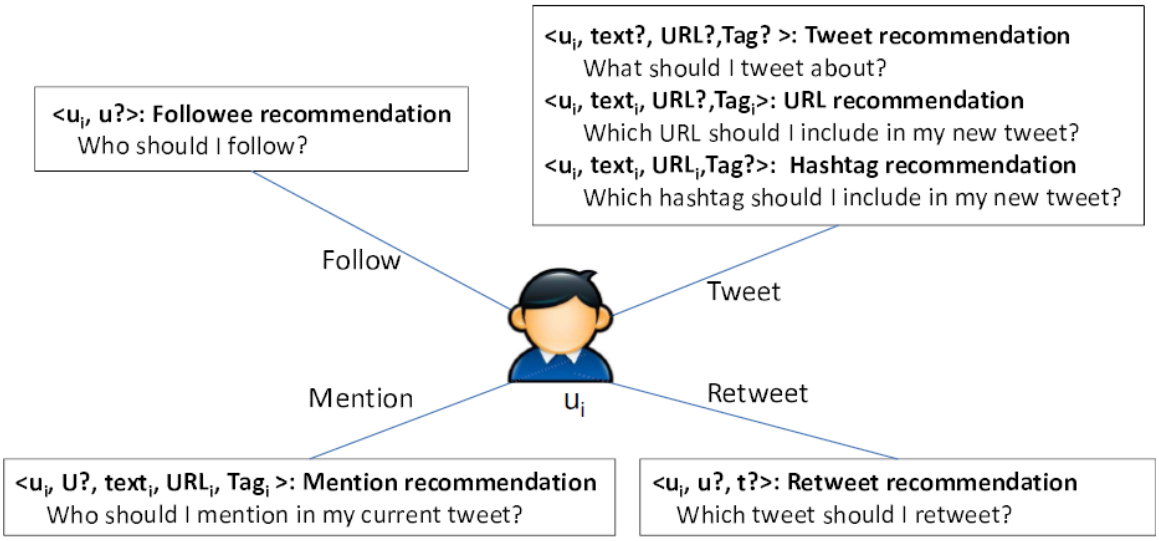
Function	Tuple	Function	Tuple
$tweet_i$	$\langle u_i, text_i, Url_i, Tag_i \rangle$ u_i: user who tweets $text_i$: tweet's text Url_i: set of URLs in the tweet Tag_i: set of tags in the tweet	$mention_i$	$\langle u_i, U_i, text_i, Url_i, Tag_i \rangle$ u_i: user who mentions others U_i: users who are mentioned $text_i$: the tweet's text Url_i: set of URLs in the tweet Tag_i: set of tags in the tweet
$retweet_i$	$\langle u_i, u_j, t_j \rangle$ u_i: user who retweets u_j: user whose tweet is retweeted t_j: the tweet that is retweeted (URLs and tags may already exist in t_j)	$follow_i$	$\langle u_i, u_j \rangle$ u_i: user who follows u_j: user who is followed

Πίνακας 2.2 : Πώς ορίζονται οι λειτουργίες του Twitter

Το Twiter όπως και το «Rayzit», αλλά και το Facebook, το Ebay και το Amazon είναι υπηρεσίες που διαχειρίζονται Μεγάλα Δεδομένα (Big - Data). Με τον όρο Μεγάλα δεδομένα γίνεται η αναφορά σε δεδομένα τα οποία έχουν πολύ μεγάλο μέγεθος και τις πλείστες των περιπτώσεων είναι τόσο μεγάλο που χρειάζονται παράλληλες ή/και κατανεμημένες προσεγγίσεις για την ανάλυση τους. Η σχεδίαση και υλοποίηση ενός συστήματος συστάσεων εμπίπτει σε αυτή την κατηγορία λόγω του μεγάλου μεγέθους δεδομένων που πρέπει να τύχουν επεξεργασίας.

Όταν γίνεται λόγος για μεγέθη συνήθως εννοούμε από Terrabytes μέχρι και Petabytes. Σύγχρονοι ορισμοί καθορίζουν ότι ο ορισμός των μεγάλων δεδομένων δίνεται από το 3V, όπου είναι Volume - Velocity – Variety. Με τον όρο Volume δηλώνεται ο τεράστιος όγκος των δεδομένων που παράγονται ενώ το Velocity δηλώνει την μεγάλη ταχύτητα με την οποία συμβαίνει αυτό. Και το Variety δηλώνει ότι τα δεδομένα δεν έχουν σταθερή δομή κι μπορεί να είναι οτιδήποτε από πολυμέσα μέχρι και μη δομημένα δομημένα.

Στη συγκεκριμένη Ατομική Διπλωματική Εργασία το θέμα που έχει αναφερθεί θα μας απασχολήσει αρκετά για να προταθεί μια λύση που δεν θα υποφέρει από τεράστιους χρόνους εκτέλεσης. Γενικότερα σε προβλήματα που εμπίπτουν στη κατηγορία των Μεγάλων Δεδομένων οι προσεγγίσεις που γίνονται πρέπει να είναι προσεγμένες και ανεπτυγμένες με τρόπο που αποδίδουν στο μέγιστο.



Πίνακας 2.3 : Πώς ορίζονται εικονικά οι λειτουργίες του Twitter

Αυτό φαίνεται ερευνώντας το Twitter. Κάθε μέρα πρέπει να διαχειρίζεται και να αποθηκεύει γύρω στα 300 εκατομμύρια tweets. Το Twitter δηλαδή πρέπει να χρησιμοποιήσει ένα τρόπο που να μπορεί γρήγορα να επεξεργάζεται αυτό το τεράστιο όγκο δεδομένων και να μπορεί έτσι να προσφέρει στο χρήστη που το χρησιμοποιεί τα κατάλληλα εργαλεία για να κάνει τις κατάλληλες επιλογές που είναι οι καλύτερες για αυτόν.

Μέσα από τη βιβλιογραφία που μελετήθηκε έχουν βρεθεί αρκετά δημοσιευμένα άρθρα που παρουσιάζουν έρευνα πάνω στο τρόπο που το Twitter υλοποιεί συστήματα σύστασης χωρίς ωστόσο να παρουσιάζεται ο τρόπος που το κάνουν. Η βιβλιογραφία που έχει βρεθεί παρουσιάζει το Twitter σε καθαρά ερευνητικό επίπεδο [1] .

Το Twitter ταξινομείται στις ακόλουθες λειτουργίες που υποστηρίζει στην υπηρεσία του:

1. Tweet
2. Retweet
3. Follow
4. Mention

Στο πίνακα 2.1 παρουσιάστηκε η ταξινομία που παρέχεται στο Twitter. Από τον ορισμό των λειτουργιών του Tweet όπως φαίνεται στο Πίνακα 2.2 και 2.3 γίνεται αντιληπτό ότι για ένα χρήστη (user) όταν θα θελήσει να στείλει π.χ ένα tweet, ο σκοπός ενός συστήματος συστάσεων είναι να τον βοηθήσει τι πρέπει να βάλει στη λίστα με τις διευθύνσεις URL και αντίστοιχα στα Tags. Άλλο παράδειγμα είναι αυτό του Mention. Ο χρήστης θα ήθελε το σύστημα να του προτείνει σε σχέση με αυτά που δακτυλογραφεί κάποια hashtags ή ακόμα

και χρήστες. Ουσιαστικά ο χρήστης πολλές φορές ξέρει τι θέλει να πει αλλά δεν ξέρει ποια URL και Tags να χρησιμοποιήσει.

Τα συστήματα σύστασης χωρίζονται σε εξατομικευμένα (personalized) και μη-εξατομικευμένα (non - personalized). Τα μη – εξατομικευμένα συστήματα δεν κάνουν χρήση των ενδιαφερόντων των χρηστών. Γι' αυτό και δεν θα μας απασχολήσουν. Από την άλλη τα εξατομικευμένα είναι αυτά που λαμβάνουν υπόψη τα ενδιαφέροντα των χρηστών.

Συγκεκριμένα χωρίζονται σε δύο κατηγορίες:

- Collaborative Filtering
- Content - Based Approaches

Στο Collaborative Filtering υπάρχουν δύο ειδών προσεγγίσεις που χρησιμοποιούνται. Υπάρχει η προσέγγιση στην οποία θα βασιστεί στους χρήστες και θα ονομαστεί User – to – User Collaborative Filtering Approach και υπάρχει και η προσέγγιση που θα βασιστεί στα αντικείμενα των χρηστών και θα ονομαστεί Item – to – Item Collaborative Filtering Approach. Ακολουθεί πλήρης ορισμός των δύο προσεγγίσεων.

Το User – to – User Collaborative Filtering Approach εφαρμόζεται στη περίπτωση που αν ένα άτομο Χ έχει την ίδια άποψη με ένα άτομο Υ πάνω σε ένα θέμα Α, τότε ο Χ είναι ποιο πιθανόν να υιοθετήσει την άποψη του Υ, σε ένα διαφορετικό θέμα Β, σε σχέση με ένα άλλο τυχαίο άτομο. Αυτή η προσέγγιση βρίσκει χρήστες που χρησιμοποιούν την υπηρεσία με κοινά ενδιαφέροντα σε σχέση με το παρελθόν τους. Το παρελθόν τους μπορεί να είναι αγορές, μηνύματα και ούτω καθεξής.

Από την άλλη το Item – to – Item Collaborative Filtering Approach εφαρμόζεται όταν δύο αντικείμενα Α και Β είναι σε μεγάλο βαθμό όμοια αν ένα μεγάλο ποσοστό των χρηστών που αγόρασαν το Α, στη συνέχεια αγόρασαν και το Β. Αυτή η προσέγγιση χρησιμοποιείται κυρίως στο Amazon η οποία και θα επεξηγηθεί στην επόμενη υποενότητα.

Η μέθοδος Content – Based Approach βρίσκει αντικείμενα που μοιάζουν μεταξύ τους, συγκρίνοντας διάφορα χαρακτηριστικά τα οποία έχουν [2,3,4]. Το σύστημα σύστασης χρησιμοποιείται για χρήστες που είτε τους άρεσαν είτε αγόρασαν παρόμοια αντικείμενα στο παρελθόν. Σε αυτή τη περίπτωση πολλά υποψήφια αντικείμενα συγκρίνονται με αντικείμενα που προηγουμένως έχουν αρέσει – αγοραστεί από τους χρήστες και αυτά που είναι ποιο κοντά σε αυτά προτείνονται πάντα σε σχέση με τα χαρακτηριστικά που τα διέπουν.

Το σύστημα σύστασης στο Twitter και συγκεκριμένα στην Who to follow περίπτωση, θα προτείνει άτομα για να ακολουθήσει (follow) κάποιος χρήστης σε σχέση με τα άτομα που παρουσιάζουν ομοιότητα με τα άτομα που ήδη ακολουθεί (according to his followees) αλλά και με τα άτομα που ακολουθούν αυτοί που ακολουθούν (according to the followees of those he follows). Αυτό γίνεται όταν κάποιος χρήστης επισκεφτεί το προφίλ ενός άλλο χρήστη, άλλοι χρήστες που είναι παρόμοιοι με αυτό το προφίλ θα συσταθούν σε αυτόν. Ο αλγόριθμος που επιτελεί αυτή την εργασία είναι άγνωστος στο ευρύ κοινό. Συχνά εταιρείες που θέλουν να διαφημιστούν πληρώνουν το Twitter, αλλά και το Facebook, και με αυτό τον τρόπο προωθούνται στις διαφημίσεις ως συστάσεις προς το κοινό. Αυτοί στην αγγλική ορολογία ονομάζονται «promoted accounts».

Στο Collaborative Filtering Approach χρησιμοποιείται ένας αλγόριθμος τοπολογίας και οι όμοιοι χρήστες βρίσκονται με βάση ένα γράφο ακολουθιών (follow graph). Ο τελικός χρήστης που θα προταθεί σε κάποιο άλλο χρήστη πρέπει να είναι όμοιος με αυτόν. Οι υποψήφιοι χρήστες για κάποιο χρήστη κατατάσσονται ανάλογα με το πόσους κοινούς άλλους χρήστες ακολουθούν. Επιλέγονται αυτοί που έχουν τους περισσότερους κοινούς. Με αυτό εξασφαλίζεται ότι για να ακολουθούν πολλούς κοινούς χρήστες σημαίνει ότι τείνουν να έχουν κοινά ενδιαφέροντα. Από την άλλη στο Content – Based Approach μας ενδιαφέρει το περιεχόμενο που ανταλλάσσεται μεταξύ των χρηστών. Γι' αυτό τα ενδιαφέροντα των χρηστών λαμβάνονται υπόψη από το περιεχόμενο των tweet. Σε αυτή τη περίπτωση αυτοί που ακολουθεί κάποιος χρήστης, αν τα tweets τους είναι όμοια με αυτών που ακολουθούν εκείνοι τότε αυτοί που ακολουθούν πολύ πιθανόν να έχουν κοινά ενδιαφέροντα με τον χρήστη που εξετάζεται άρα και θα είναι πολύ πιθανόν να προταθούν σε αυτόν. Κοινώς ισχύει ότι ο χρήστης θα ακολουθεί αυτούς που είναι όμοιοι με αυτόν.

Άλλες προσεγγίσεις όπως οι Weighted Content – Based Methods [5], είναι οι συστάσεις να είναι ανάλογα με τη δημοτικότητα (popularity), τις δραστηριότητα που δείχνε το προφίλ κάποιου χρήστη, τη τοποθεσία, ποιους κοινούς φίλους έχει με κάποιο άλλο άτομο. Ποιο συγκεκριμένα η δημοτικότητα μετριέται με ένα ποσοστό σε σχέση με αυτούς που ακολουθεί και αυτούς που τον ακολουθούν. Η δραστηριότητα από την άλλη μετριέται σε σχέση με τον αριθμό των tweet που κάποιος χρήστης έχει στείλει από την ημέρα που καταχωρήθηκε ως ενεργός χρήστης στην υπηρεσία. Το πρόβλημα με αυτές τις λύσεις είναι ότι αν κάποιος χρήστης είναι όντως δημοφιλής τότε το σύστημα θα του συνιστά άλλους χρήστες που και εκείνοι είναι δημοφιλής και έχουν μεγάλη δραστηριότητα στην υπηρεσία. Αυτό σημαίνει ότι αν 10 χρήστες έχουν μεγάλο ενδιαφέρον για ένα ενδιαφέρον Χ τότε το σύστημα για άλλους 5 διαφορετικούς χρήστες που επίσης αρέσει το Χ, θα τους προτείνει όλους τους 10 που

προαναφέρθηκαν και με αυτό τον τρόπο το σύστημα σύστασης δεν θα προτείνει ανερχόμενους χρήστες της υπηρεσίας και άρα θα επιβραβεύει μόνο τους υφιστάμενους.

Ακόμα η προσέγγιση Followee Recommendation – Structural Methods που προτείνει χρήστες σε κάποιο χρήστη για να τους ακολουθήσει βασίζεται πάνω στο ότι ένας χρήστης Α πολύ πιθανόν να ακολουθήσει κάποιο χρήστη Β, αν ο χρήστης Β τον έχει ακολουθήσει προηγουμένως απλά και μόνο για να ανταποδώσουν το ενδιαφέρον που έδειξε ο Β. Ο ορολογία στην αγγλική γλώσσα ονομάζεται «reciprocity» [6]. Σύμφωνα με τον ορισμό που δόθηκε προηγουμένως ένα σύνολο από χρήστες θεωρούνται όμοιοι αν ακολουθούν τους ίδιους χρήστες. Έτσι το σύστημα συστάσεων θα προτείνει σε ένα χρήστη τους χρήστες που τον ακολουθούν και αυτό όχι.

Κάτι αντίστοιχο συμβαίνει και στη περίπτωση που δύο χρήστες ακολουθούν τους ίδιους χρήστες αλλά οι ίδιοι ο ένας δεν ακολουθεί τον άλλο. Βασιζόμενο πάνω στο ότι αν δυο χρήστες ακολουθούν ίδιους χρήστες τότε θα είναι και οι ίδιοι όμοιοι το σύστημα συστάσεων θα προτείνει τον ένα στον άλλο.

Το σύστημα συστάσεων που βασίζεται στα HashTags [7], αντιλαμβάνεται ότι αν ένα σύνολο από hashtags συνδέεται με αρκετά προφίλ χρηστών τότε αυτό δείχνει ότι τα hashtags αυτά αντικατοπτρίζουν τα ενδιαφέροντα των χρηστών. Άρα το σύστημα συστάσεων θα προτείνει στους χρήστες άλλους χρήστες οι οποίοι έχουν κοινά hashtags. Επίσης οι αλγόριθμοι που εξετάζουν τέτοιες προσεγγίσεις θα εξετάσουν και αυτούς που ακλουθούν αυτούς που ακολουθούν οι χρήστες και αν έχουν κοινά hashtags – ενδιαφέροντα θα προταθούν.

Άλλοι αλγόριθμοι θα προτείνουν στους χρήστες άλλα hashtags τα οποία μπορούν να χρησιμοποιήσουν σε σχέση με το περιεχόμενο του tweet που δημιουργούν σε πραγματικό χρόνο και σε σχέση με τα hashtags χωρίς να λαμβάνουν υπόψη το παρελθόν του χρήστη όσων αφόρα τα ενδιαφέροντα του στα hashtags. Από την ώρα που ο χρήστης θα αρχίσει να πληκτρολογεί το σύστημα βρίσκει άλλα tweets που είναι όμοια με αυτό και προτείνει πιθανά hashtags που είναι σε μεγάλο βαθμό όμοια με το θέμα που γράφει ο χρήστης άλλα και επίσης που έχουν αρκετή δημοτικότητα (έχουν χρησιμοποιηθεί σε μεγάλο βαθμό από άλλους χρήστες) σε άλλα tweets.

2.3.2Amazon και Συστήματα Συστάσεων

Στις πλατφόρμες αγοραπωλησίας Amazon και Ebay, λόγω της διαφορετικού τύπου υπηρεσίας που προσφέρουν τα συστήματα σύστασης δουλεύουν με διαφορετικό τρόπο.

Χρησιμοποιούνται αλγόριθμοι που προσπαθούν να εξάγουν τα ενδιαφέροντα των χρηστών σε σχέση με αυτά που αγοράζουν αλλά και που είδαν. Τα προβλήματα που έχουν σε σχέση με τα συστήματα συστάσεων είναι στους καινούργιους χρήστες έχουν συγκεντρωμένα από λίγα έως και καθόλου στοιχεία για αυτούς. Αλλά και για παλαιούς χρήστες που έχουν πολλά ενδιαφέροντα θα γίνεται η σύσταση σε σχέση με όλα τα ενδιαφέροντα και όχι για τα ποιο πρόσφατα.

Οι βασικές προσεγγίσεις που λύουν τα ποιο πάνω προβλήματα είναι τύπου Collaborative Filtering [9] και ουσιαστικά το τι κάνουν είναι να βρίσκουν στους χρήστες παρόμοιους άλλους χρήστες που αγόρασαν σχετικά με αυτούς αντικείμενα στο παρελθόν και τους προτείνουν προϊόντα που ακόμα δεν έχουν αγοράσει. Αλγοριθμικά όμως η λύση αυτή κοστίζει γιατί θα πρέπει να περάσει πάνω από όλους τους χρήστες και τα αντικείμενα που έχουν αγοράσει. Υπάρχουν αρκετές παραλλαγές που π.χ επιλέγουμε τυχαίο δείγμα από χρήστες που θα λάβουν μέρος στη διαδικασία, ή να απορρίψουμε χρήστες με λίγες αγορές. Ακόμα, κάποια άλλα παραδείγματα είναι να απορρίψουμε δημοφιλή και μη δημοφιλή αντικείμενα ή να περιορίσουμε τα δεδομένα κάνοντας ενός είδους ομαδοποίηση. Έτσι με αυτό τον τρόπο θα απαιτηθεί ποιο λίγος χρόνος για υπολογισμό.

Οι ποιο πάνω λύσεις θα επιφέρουν αρκετά προβλήματα στη ποιότητα της σύστασης. Αρχικά να λεχθεί ότι δεν είναι σωστό να εξεταστεί μόνο μια μικρή μερίδα χρηστών αφού υπάρχει κίνδυνος αυτοί που στο τέλος θα επιλεχθούν ως οι ποιο κοντά σε κάποιο χρήστη να μην είναι και τόσο αντιπροσωπευτικά κοντά του. Από την άλλη αν απορρίψουμε δημοφιλή ή μη δημοφιλή αντικείμενα πάλι προκύπτει πρόβλημα αφού αυτά τα αντικείμενα μας ενδιαφέρουν και δεν θα εμφανίζονται ποτέ ως υποψήφια αντικείμενα για σύσταση.

Άλλες προσεγγίσεις χρησιμοποιούν Μοντέλα Ομαδοποίησης (Cluster Models). Τα συστήματα αυτά βρίσκουν χρήστες που είναι όμοιοι με το χρήστη. Για να το κάνουν αυτό ομαδοποιούν τους χρήστες με βάση τα ενδιαφέροντα που έδειξαν σε αυτά που έχουν αγοράσει. Όμως η σύσταση που προκύπτει είναι χαμηλή γιατί με τον τρόπο που γίνονται τα μοντέλα ομαδοποίησης ο διαχωρισμός που κάνουν για τους χρήστες δεν αντικατοπτρίζει πλήρως τα ενδιαφέροντα τους. Αναθέτουν τους χρήστες σε μια συστάδα (cluster) σε σχέση με αυτά που έχει αγοράσει. Αυτό δεν δουλεύει πάντα καλά. Έτσι με αυτό τον τρόπο μπορεί να ανατεθεί σε μια συστάδα που δεν είναι κοντά στις προτιμήσεις του. Άρα οι συστάσεις που θα προκύψουν δεν θα είναι αντιπροσωπευτικές.

Άλλες προσεγγίσεις που χρησιμοποιούνται στο Amazon, λαμβάνουν υπόψη τις αναζητήσεις που κάνει ένας χρήστης. Γι' αυτό και τα διάφορα συστήματα θα προτείνουν άλλα αντικείμενα που είναι κοντά στο είδος των αντικειμένων που ο χρήστης αναζητεί χρησιμοποιώντας την υπηρεσία. Αυτή η προσέγγιση δουλεύει καλά αν ο χρήστης έχει στο ιστορικό του αναζητήσεις. Εντούτοις η γενική εικόνα που βγάζει αυτή η προσέγγιση δεν είναι καλή και η σύσταση είναι πολύ χαμηλής ποιότητας.

Αυτό που χρησιμοποιείται κατά κύριο λόγο είναι το Item – to – Item Collaborative Filtering όπως αυτό επεξηγήθηκε στη προηγούμενη ενότητα για το Twitter. Ουσιαστικά θα ληφθεί υπόψη το ιστορικό του χρήστη σε σχέση με τα αντικείμενα που αγόρασε και έδωσε βαθμολογία. Στη συνέχεια ο αλγόριθμος θα βρει χρήστες που έχουν αγοράσει παρόμοια προϊόντα με αυτόν και θα του προτείνει προϊόντα που αυτοί έχουν αγοράσει και αυτός όχι.

Κεφάλαιο 3

Μοντέλο Συστήματος

Αρχικά σε αυτό το κεφάλαιο θα δοθεί ένα γενικό περίβλημα που συνοψίζει τη βιβλιογραφία γύρω από το πρόβλημα του kNN και θα αναφερθούν κάποια προβλήματα που καλείται να λύσει σε γενικό επίπεδο και ο δικός μας αλγόριθμος. Ακολούθως θα δοθεί ο καθορισμός του προβλήματος καλείται να λύσει η συγκεκριμένη Ατομική Διπλωματική Εργασία. Στο τέλος θα γίνει αναφορά για τις διάφορες παραδοχές που έγιναν για τη λύση του προβλήματος και στο τέλος θα δοθεί ένα γενικό μοντέλο συστήματος που θα βοηθήσει στην αξιολόγηση των προσεγγίσεων – αλγορίθμων που θα παρουσιαστούν στο επόμενο κεφάλαιο.

3.1 Αλγόριθμοι Εγγύτητας

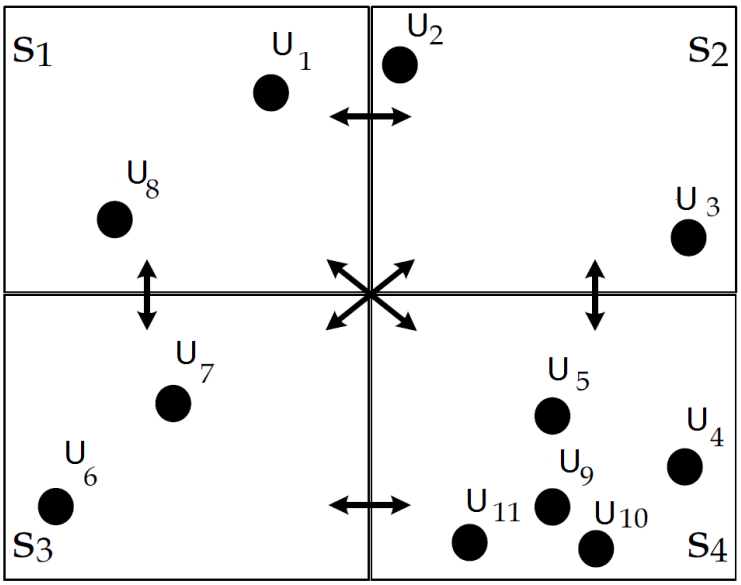
Σε αυτή την ενότητα θα αναφερθούν κάποιοι τεχνικοί όροι που αφορούν το αντικείμενο των k κοντινότερων γειτόνων (kNN). Θα επεξηγηθεί πώς ο όρος kNN, βρίσκει εφαρμογή στη διπλωματική αυτή εργασία. Για να γίνει αντιληπτό ένα μέρος του προβλήματος που πρέπει να λυθεί θα πρέπει να επεξηγηθεί το kNN και στη συνέχεια το AkNN.

3.2 k – Κοντινότεροι Γείτονες (k – Nearest Neighbors - kNN) και Παραλλαγές

Οι k κοντινότεροι γείτονες [9,10] (k – Nearest Neighbors - kNN) για ένα αντικείμενο ο μέσα σε ένα σύνολο δεδομένων O, το οποίο δηλώνεται ως kNN(o,O) είναι τα k αντικείμενα που είναι ποιο «κοντά» στις προτιμήσεις και στα χαρακτηριστικά του αντικειμένου ο. Γι' αυτό και το «κοντά» μπορεί να οριστεί είτε ως πραγματική απόσταση σε ένα χώρο, είτε ενδιαφέροντα, κτλ. Δοθέντος κάποιων αντικειμένων έστω $o_a \neq o_b \neq o_c$ και $\forall\, o_b \in kNN(o_a, O)$ και $\forall\, o_c \in O - kNN(o_a, O)$ ισχύει πάντοτε ότι $dist\,(o_a, o_b) < dist\,(o_a, o_c)$. Το πρόβλημα του kNN είναι ένα κλασσικό πρόβλημα που απασχολεί τη κατεύθυνση της Πληροφορικής με αρκετές εντατικοποιημένες λύσεις που βρίσκουν εφαρμογές σε προβλήματα που αφορούν υπολογιστική γεωμετρία, επεξεργασία εικόνας, χωρικές βάσεις δεδομένων (spatial databases), κοινωνικά δίκτυα (social networks), για ανάλυση και επεξεργασίας δεδομένων εκτός πραγματικού χρόνου (offline analytics) όπως είναι η ομαδοποίηση (clustering), αναζήτηση ψευδών δεδομένων (outliers).

Το AkNN [11,12] (All k – Nearest Neighbors) είναι μια γενίκευση του υπάρχοντος kNN προβλήματος. Η διαφορά έγκειται στο γεγονός ότι σε αυτή τη περίπτωση θέλουμε να λύσουμε το kNN(o,O) πρόβλημα για κάθε αντικείμενο $o \in O$. Στη περίπτωση μας είναι προφανές ότι το σύνολο O που δίνεται από τον ορισμό αντικαθιστάται από το σύνολο U που είναι οι χρήστες του συστήματος. Θέλουμε δηλαδή να υπολογίζουμε του k ποιο κοντινούς γείτονες του κάθε χρήστη $u_i \in U$.

Υπάρχουν αρκετές προσεγγίσεις που προσπαθούν να λύσουν το AkNN. Αρκετές από αυτές είναι οι βέλτιστες δυνατές και είναι ειδικευμένες για επεξεργασίας δεδομένων εκτός πραγματικού χρόνου και οι λύσεις τους βασίζονται στο ότι τα δεδομένα βρίσκονται στη μνήμη RAM (memory – resident data), ή μέσα σε κοινή μνήμη (shared memory) ή ακόμα και σε δευτερεύουσες πηγές αποθήκευσης.



Σχήμα 3.1 : Σενάριο εύρεσης 2-NN για κάθε χρήστη για τη συγκεκριμένη διάταξη χρηστών

3.3 Προκλήσεις και εφαρμογές Λύσεων της Πλατφόρμας Rayzit

Το πρώτο πρόβλημα που καλείται ένας οποιοσδήποτε κατανεμημένος αλγόριθμος να αντιμετωπίσει είναι το πρόβλημα της σωστής κατανομής των δεδομένων. Επίσης ένα άλλο πρόβλημα είναι και αυτό του κόστους επικοινωνίας μεταξύ των διαφόρων εξυπηρετητών που ανταλλάζουν κατάσταση μεταξύ τους για να γίνει ο σωστός υπολογισμός του kNN. Ας υποθέσουμε το εξής σενάριο που μπορούμε να εξετάσουμε με εικονική αναπαράσταση στο Σχήμα 3.6. Στο σχήμα φαίνονται οι διάφοροι χρήστες που βρίσκονται κατανεμημένοι στο δυσδιάστατο χώρο.

Πρόβλημα Κόστους Επικοινωνίας

Στο Σχήμα 1.4 υπάρχουν 11 χρήστες οι οποίοι διαχωρίζονται σε 4 χωρικά ισοδύναμους εξυπηρετητές $\{ s_1 \dots s_4 \}$. Ας υποθέσουμε για χάριν παραδείγματος ότι θέλουμε σαν αποτέλεσμα το 2-NN για κάθε χρήστη. Τότε οπτικά μπορούμε να παρατηρήσουμε το 2NN $(u_1, U) = (u_2, u_8)$. Σε αυτό το παράδειγμα μπορούμε να διακρίνουμε ότι ο χρήστης 8 βρίσκεται μέσα στον ίδιο χώρο με τον χρήστη 1 κάτι το οποίο δεν ισχύει για το χρήστη 2. Άρα για να υπολογιστεί η απόσταση $dist\,(u_1, u_2)$ θα πρέπει πρώτα να μεταφέρουμε το χρήστη 2 από τον s_2 στον s_1. Το ίδιο προκύπτει και για άλλους χρήστες όπως το 2NN $(u_8, U) = (u_7, u_1)$, 2NN $(u_6, U) = (u_7, u_8)$.

Πρόβλημα Φόρτου Εργασίας Εξυπηρετητών

Ένα άλλο πρόβλημα που αναφέρθηκε μερικώς προηγουμένως είναι αυτό της ισότιμης κατανομής φόρτου εργασίας στους εξυπηρετητές. Για να εξαχθεί το AkNN αποτέλεσμα πρέπει να γίνει ισότιμη κατανομή χρηστών σε όλους τους s_i εξυπηρετητές. Με αυτό τον τρόπο στον κάθε s_i εξυπηρετητή θα απαιτείται ο ίδιος χρόνος για να υπολογιστούν οι διάφορες αποστάσεις για κάποιο χρήστη u_a στους διάφορους άλλους χρήστες δεδομένου ότι η απόσταση $dist\ (u_a, u_b)$ έχει σταθερό κόστος επεξεργασίας. Οπτικά αυτό μπορεί να γίνει αντιληπτό μελετώντας το Σχήμα 3.6 στο οποίο βλέπουμε ότι ο s_4 εξυπηρετητής θα πρέπει να υπολογίσει 30 αποστάσεις μεταξύ 6 χρηστών (για τους τοπικούς χρήστες $\{u_4, u_5, u_9, u_{10}, u_{11}\}$ και για το μεταφερόμενο αντικείμενο $\{u_3\}$), ενώ ο εξυπηρετητής s_3 θα πρέπει να υπολογίσει μόλις 6 αποστάσεις μεταξύ 3 χρηστών (για τους τοπικούς χρήστες $\{u_6, u_7\}$ και για το μεταφερόμενο αντικείμενο $\{u_8\}$). Αυτή η ασυμμετρία έχει σαν αποτέλεσμα ο εξυπηρετητής s_3 να τελειώσει τους υπολογισμούς που πρέπει να κάνει 5 φορές γρηγορότερα από τον εξυπηρετητή s_4. Μάλιστα ο s_4 θα καθυστερήσει αρκετά τον όλο υπολογισμό σε σχέση με τους υπόλοιπους εξυπηρετητές s_i.

Συμπέρασμα για τα δύο προβλήματα που αναφέρθηκαν

Το συμπέρασμα που προκύπτει είναι ότι για να λυθεί το πρόβλημα του AkNN θα έπρεπε να ληφθούν πολύ σοβαρά υπόψη τα δύο προβλήματα που αναφέρθηκαν γιατί η λύση αυτών των δύο προβλημάτων είναι που δίνει την επεκτασιμότητα, απόδοση και μείωση του κόστους επικοινωνίας στον αλγόριθμο SNA που θα παρουσιαστεί σε επόμενο κεφάλαιο ο οποίος θα κάνει χρήση των τεχνικών υπολογισμού του kNN.

3.4 Καθορισμός Προβλήματος

Αφού έχει επεξηγηθεί η πλατφόρμα «Rayzit» και έχουν δοθεί αρκετές κατευθυντήριες γραμμές για τη χρήση και την εφαρμογή της υπηρεσίας θα πρέπει να δοθεί η κεντρική ιδέα της συγκεκριμένης διπλωματικής εργασίας. Έχοντας υπόψη τη παρακίνηση που δόθηκε στο πρώτο κεφάλαιο να επαναληφθεί ότι ο σκοπός είναι η αύξηση της ποιότητας υπηρεσίας της πλατφόρμας Rayzit με τρόπο που να ωθήσει περισσότερους χρήστες να τη χρησιμοποιήσουν και εν τέλει να γίνει περισσότερο δημοφιλής στο κοινό.

Έχοντας μελετήσει αρκετά το πρόβλημα με τους k κοντινότερους γείτονες όπως αναφέρθηκε συνοπτικά στη προηγούμενη υποενότητα, έγινε μια σκέψη η καλυτέρευση της υπηρεσίας να συνεχίσει να γίνεται έχοντας σαν βάση το AkNN γιατί αυτό είναι αυτό που κάνει το «Rayzit» μοναδικό σε σχέση με άλλες υπηρεσίες.

Οπόταν έχοντας σαν βάση το AkNN θα έπρεπε να γίνει μια προσέγγιση που θα καλυτέρευε την ποιότητα υπηρεσίας που παρέχεται στους χρήστες. Γι' αυτό και στο δεύτερο κεφάλαιο έγινε μια έρευνα που είχε σκοπό να καταδείξει διάφορα συστήματα και πως αυτά επιτυγχάνουν να δίνουν στους χρήστες τους διάφορες «συμβουλές» ως προς το ποιο θα ακολουθήσουν (who to follow - Twitter) ,ή ποιο βιβλίο θα αγοράσουν (item based recommendations - Amazon) , ή ακόμα και ποιοι χρήστες είναι κοντά στα ενδιαφέροντα τους (user-to-user recommendation – Twitter, Amazon, Facebook).

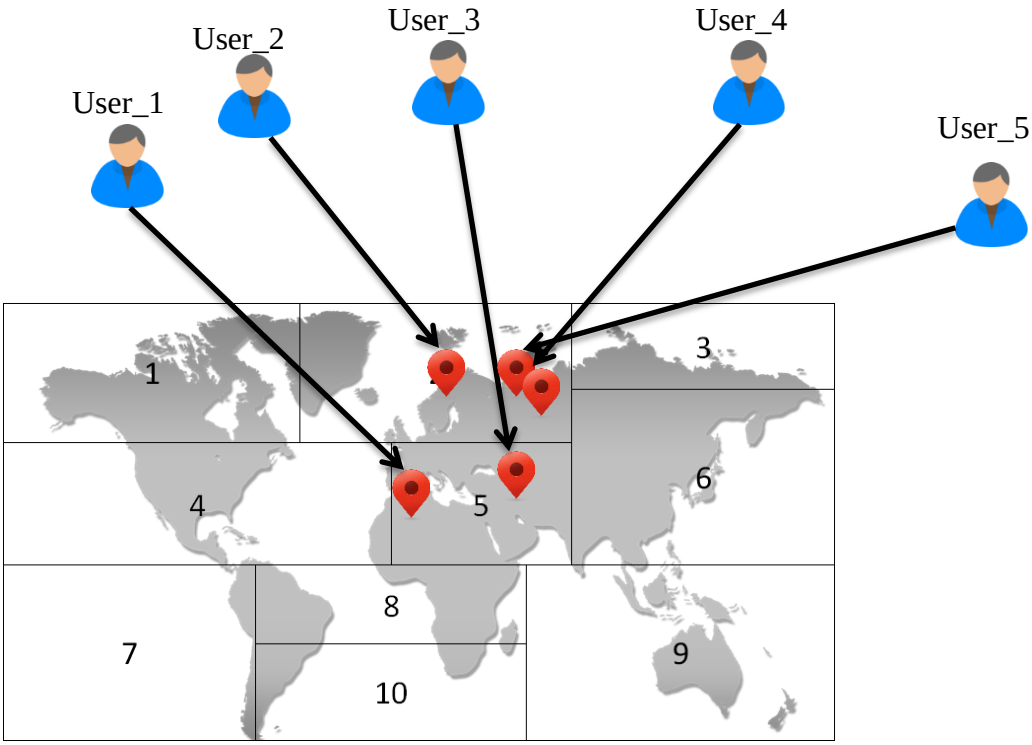
Έτσι λήφθηκε η απόφαση να γίνει η σύσταση χρηστών σε χρήστες (user-to-user recommendation) το οποίο χρησιμοποιείται κατά κύριο λόγο και από το Facebook και το Twitter που είναι υπηρεσίες κοινωνικής δικτύωσης αλλά και από την Amazon, που τα τελευταία χρόνια όπως έχουν καταδείξει οι διάφορες έρευνες έχει αυξήσει κατά πολύ την ποιότητα υπηρεσίας που προσφέρει στους πελάτες - χρήστες της.

Αυτού του είδους η σύσταση θα φέρει κοντά χρήστες σε άλλους χρήστες ανάλογα με τα κοινά τους ενδιαφέρονται όπως αυτά εκφράζονται μέσα από αυτά που λένε όταν χρησιμοποιούν την υπηρεσία «Rayzit». Θα χρησιμοποιηθούν όλα τα μηνύματα που ανταλλάζουν οι χρήστες μεταξύ τους με σκοπό ο αλγόριθμος που θα παρουσιαστεί στο επόμενο κεφάλαιο να μπορεί να αναγνωρίσει ομοιότητες μεταξύ των χρηστών. Πλήρης ορισμός του προβλήματος ακολουθεί ποιο κάτω.

User u_i
Rayz 1 : I have an android smartphone but it always freezes.
Rayz 2 : Does anyone interested in cars?
……

User u_j
Rayz 1 : Recent studies have shown that the android smartphones are very promising
Rayz 2 :I have just bought my brand new car!
…..

Σχήμα 3.2 : Δύο πιθανοί χρήστες και ρεαλιστικά μηνύματα τα οποία ανταλλάζουν καθώς χρησιμοποιούν το «Rayzit»



Σχήμα 3.3 : Στο σχήμα φαίνονται 5 χρήστες του Rayzit καθώς γεωτοποθετούνται στο χώρο

Έτσι έχει αποφασιστεί η αύξηση της ποιότητας υπηρεσίας του Rayzit να γίνεται με τον εξής τρόπο

«Να φέρουμε κοντά σε κάποιο χρήστη όχι μόνο τους k χρήστες, που είναι κοντά του γεωγραφικά, αλλά και που έχουν σε αρκετά μεγάλο βαθμό κοινά ενδιαφέροντα όπως αυτά διαφαίνονται μέσα από αυτά που λένε στα μηνύματα που ανταλλάζουν όταν χρησιμοποιούν το Rayzit»

Από το ποιο πάνω σχήμα ορίζεται ως $U = \{u_1, u_2, ..., u_n\}$ *το σύνολο των n χρηστών που χρησιμοποιούν τη πλατφόρμα «Rayzit».* Όπως προαναφέρθηκε και προηγουμένως όλοι οι χρήστες που χρησιμοποιούν τη ποιο πάνω πλατφόρμα αναγκαστικά θα πρέπει να δίνουν τη

τρέχουσα τοποθεσία τους μέσα στο χώρο για να μπορούν να χρησιμοποιήσουν όλες τις υπηρεσίες που προσφέρονται. Γι' αυτό όπως παρατηρείται και στο Σχήμα 3.3 ο κάθε χρήστης έχει καταχωρημένη τη τρέχουσα τοποθεσία του μέσα στο σύστημα.

Η τοποθεσία που έχει ο κάθε χρήστης μέσα στο σύστημα είναι πολύ σημαντική και παίζει μεγάλο ρόλο για τις διάφορες λειτουργίες και υπολογισμούς που επιτελούνται. Γι' αυτό θα ορίσουμε ως lon_{u_i} *και* lat_{u_i} *τη τελευταία γνωστή τοποθεσία του χρήστη* u *μέσα στο δυσδιάστατο χώρο που επικαλύπτει το κόσμο.*

Σε αυτό το σημείο θα πρέπει να οριστεί *ένας αριθμός* k *που είναι πληθικός αριθμός του συνόλου* K_{u_i}*, των κοντινότερων γειτόνων για ένα χρήστη* u_i *(* $K_{u_i} = |k|$ *). Ορίζουμε επίσης το* s_i *ως τον* i *– οστό εξυπηρετητή για τον οποίο ισχύει ότι* $s_i \in S$ *όπου* $|S| = m$*.*

Στο Σχήμα 3.2 φαίνεται το πρόβλημα που θέλουμε να λύσουμε. Οι δύο αυτοί χρήστες είναι πιθανόν μέσα από αυτά που έχουν διοχετεύσει στην υπηρεσία «Rayzit» να έχουν αρκετά κοινά ενδιαφέροντα έτσι όταν θα ζητήσουν από την υπηρεσία να τους στείλει τα τελευταία μηνύματα rayz που έχουν ειπωθεί να τους στείλει μηνύματα χρηστών που είναι «κοντά» τους, τόσο σε σχέση με τη γεωγραφική τους τοποθέτηση, όσο και σε σχέση με τα ενδιαφέροντα που επιδεικνύουν. Αυτό θα φανεί σχηματικά και στο επόμενο σχήμα που παρατίθεται στη συνέχεια.

Ο χρήστης u_i *για ένα συγκεκριμένο χρονικό διάστημα* t_{u_i}*, έχοντας συγκεκριμένο* lon_{u_i} *και* lat_{u_i} *θα στείλει ένα μήνυμα* $p_{u_i,f}$ *όπου το* $p_{u_i,f} \in P_{u_i}$ *. Το* P_{u_i} *ορίζεται ως το σύνολο των μηνυμάτων που έχει συντάξει ο χρήστης* u_i *και ο δείκτης* f *δεικνύει το* f*-οστό μήνυμα του χρήστη* u_i*, όπως αυτό προκύπτει από το σύνολο* $P_{u_i} = \{ p_{u_i,1} , p_{u_i,2} , \dots , p_{u_i,f} \}$*. Από αυτή τη σχέση προκύπτει ότι* $|P_{u_i}| = F$ *και ότι* $1 \leq f \leq F$ *.*

Ορίζουμε ποιο συγκεκριμένα ότι αν ο χρήστης u_i *θα στείλει ένα rayz τότε θα διέπεται από τη συγκεκριμένη εξίσωση* $p_{u_i,f}$ *= sendRayz (* t_{u_i}*, message,* lon_{u_i}*,* lat_{u_i}*). Επομένως μπορούμε να θεωρήσουμε όλα τα rayz για ένα χρήστη σαν μια συνεχή ακολουθία από τέτοιες εγγραφές όπως αυτές επεξηγήθηκαν μέχρι τώρα δηλαδή* $P_{u_i} = \{ p_{u_i,1} , p_{u_i,2} , \dots , p_{u_i,f} \}$*.*

Ενδιαφέροντα	Χρήστες				
	User_1	User_2	User_3	User_4	User_5
Android	X	X	X	X	X
Car		X	X	X	X
Football	X	X			
Cooking	X	X	X		X

Πίνακας 3.4 : Πίνακας ενδιαφερόντων των χρηστών του Rayzit

Έχοντας δώσει τους απαραίτητους ορισμούς που αφορούν την μετέπειτα κατανόηση θα πρέπει να δοθεί ένα παράδειγμα σύμφωνα με το Πίνακα 3.4 για να γίνει πλήρως αντιληπτό το πρόβλημα που πρέπει να λυθεί

Για χάριν παραδείγματος έστω ότι θέλουμε να υπολογίσουμε το 2-NN(u_5, U). Όπως φαίνεται στο Σχήμα 4.2 για τον χρήστη 5, διαισθητικά οι 2 ποιο κοντινοί χρήστες που βρίσκονται σε ποιο μικρή πραγματική απόσταση σε σχέση με τους υπόλοιπους χρήστες μέσα στο χώρο είναι οι χρήστες 2 και 4. Όμως, αν εξετάσουμε το πίνακα 3.4 θα παρατηρήσουμε ότι «κοντά» από άποψης ενδιαφερόντων βρίσκονται οι χρήστες 2,3,4 αντίστοιχα.

Το ερώτημα λοιπόν που εγείρεται σε αυτό το σημείο είναι ποιο θα είναι τελικά το κριτήριο με το οποίο θα γίνει η επιλογή των αποτελεσμάτων. Διότι από τη μία θέλουμε να αυξήσουμε τη ποιότητα υπηρεσίας του Rayzit εισάγοντας σύστημα σύστασης αλλά από την άλλη δεν θέλουμε να εξαλείψουμε το AkNN το οποίο παίζει σημαντικό ρόλο στην λειτουργία του Rayzit.

Η απάντηση στο ποιο πάνω ερώτημα είναι πως και τα δύο θα ληφθούν υπόψη. Γι' αυτό και η προσέγγιση που θα ακολουθηθεί και θα φανεί στο επόμενο κεφάλαιο θα λαμβάνει υπόψη τις και τις δύο παραμέτρους που είναι τόσο η πραγματική Ευκλείδεια απόσταση όσο και τα ενδιαφέροντα των χρηστών της εφαρμογής. Η συγκεκριμένη προσέγγιση ονομάζεται *Multi-Objective Approach* [19,20] και είναι μια τεχνική που χρησιμοποιείται σε αλγορίθμους η οποία σαν μέτρο απόφασης χρησιμοποιεί περισσότερες από μια μετρικές. Στη περίπτωση μας οι μετρικές είναι :

(1) Ευκλείδεια Απόσταση των Χρηστών
(2) Ενδιαφέροντα Χρηστών

Καταλήγουμε στο συμπέρασμα ότι για το ποιο πάνω παράδειγμα η σωστή απάντηση για το το 2-NN(u_5, U) = { u_4 , $[u_2 \vee u_3]$}. Θα είναι σίγουρα ο χρήστης 4 διότι έχει αρκετά κοινά ενδιαφέροντα μαζί με το χρήστη 5. Ο χρήστης 4 είναι η ιδανική – ιδεατή περίπτωση στην οποία είναι μεν γεωγραφικά κοντά στον 5 αλλά βρίσκεται και κοντά στα δικά του ενδιαφέροντα. Ο ενδοιασμός που προέκυψε για την απόφαση μεταξύ του χρήστη 2 και 3 οφείλεται στο γεγονός ότι εξαρτάται από το πού θα δοθεί περισσότερο βάρος (Multi-Objective case). Αν δοθεί βάρος στην απόσταση τότε σωστή απάντηση θα είναι ο χρήστης 3, ενώ αν δοθεί στα ενδιαφέροντα τότε θα είναι ο χρήστης 2.

Ο τελικός αλγόριθμος θα ονομαστεί Αλγόριθμος Σημασιολογικής Ομοιότητας Γειτόνων (Semantic Neighbor Algorithm) και θα χρησιμοποιεί μια διαφορετική προσέγγιση του AkNN που θα ονομάζεται All k -SNN (AkSNN)

3.4 Πίνακας Συμβολισμών

Symbol	Definition
$\boldsymbol{U}$	Σύνολο των χρηστών που χρησιμοποιούν το Rayzit |U| = n
$\boldsymbol{u_i}, \boldsymbol{u_i} \in \boldsymbol{U}$	Τυχαίος χρήστης u_i όπου $u_i \in U$
$\boldsymbol{U_{s_i}}$	Οι χρήστες που ανατίθενται για επεξεργασία στον εξυπηρετητή s_i
$\boldsymbol{lon_{u_i}}$ **και** $\boldsymbol{lat_{u_i}}$	Τελευταία γνωστή τοποθεσία του χρήστη u_i μέσα στο δυσδιάστατο χώρο που επικαλύπτει τη Γη.
$\boldsymbol{K_{u_i}}$	Σύνολο με τους κοντινότερους γείτονες του χρήστη u_i
$\boldsymbol{\|K_{u_i}\| = k}$	k ορίζεται ως ο πληθικός αριθμός του συνόλου K_{u_i}
$\boldsymbol{s_i}, \boldsymbol{s_j} \in \boldsymbol{S}, \boldsymbol{\|S\| = m}$	s_i ορίζουμε ως τον ι-οστό εξυπηρετητή server για τον οποίο ισχύει $s_i \in S$ όπου για του server S ισχύει ότι το άθροισμα τους ισοδυναμεί με m
$\boldsymbol{R_{s_i}}$	R_{s_i} είναι το σύνολο που περιέχει όλα τα border candidate sets του εξυπηρετητή s_i
$\boldsymbol{R_{s_i,s_j}}, \boldsymbol{s_i} \neq \boldsymbol{s_j}$	R_{s_i,s_j} είναι οι χρήστες που στέλνονται από το τον s_i στον s_j εξυπηρετητή
$\boldsymbol{P_{u_i}} = \{\boldsymbol{p_{u_i,1}}, \boldsymbol{p_{u_i,2}}, \dots, \boldsymbol{p_{u_i,f}}\}$, $\boldsymbol{\|P_{u_i}\| = F}$	Σύνολο όλων των rayz του χρήστη u_i όπου $\|P_{u_i}\| = F$
$\boldsymbol{p_{u_i,f}}$	Τυχαίο μήνυμα f $p_{u_i,f}$ για του χρήστη u_i
$\boldsymbol{t}$	Token – Λέξη / Ενδιαφέρον που παρουσιάστηκε κατά την διάρκεια επεξεργασία των rayz όλων των χρηστών.
$\boldsymbol{D_{p_{u_i}}}$	Όλες οι πιθανές λέξεις που μπορεί προκύπτουν στο μήνυμα $p_{u_i,f}$
$\boldsymbol{H_{p_{u_i}}}, \boldsymbol{h} \in \boldsymbol{H_{p_{u_i}}}$	Όλες οι λέξεις που αντικατοπτρίζουν τα ενδιαφέροντα που προκύπτουν στο μήνυμα $p_{u_i,f}$
$\boldsymbol{T}$	Σύνολο όλων των μοναδικών tokens που παρουσιάστηκαν σε ένα εξυπηρετητή
$\boldsymbol{T_{u_i}}$	Σύνολο όλων των μοναδικών tokens του χρήστη u_i
$\boldsymbol{t} \in \boldsymbol{T_{u_i}}$	Το t είναι ένα token που ανήκει στα tokens του χρήστη u_i
$\boldsymbol{S}$	Σύνολο με token που δεν αποτελούν StopWords, δηλαδή

	λέξεων που αντικατοπτρίζουν τα πραγματικά ενδιαφέροντα του χρήστη
UserTokens	Για κάθε χρήστη u_i όπου $u_i \in U$ ποια tokens έχει αναφέρει στα μηνύματα του και πόσες φορές
TokenUsers	Για κάθε token t, όπου $t \in T_{u_i}$, φυλάσσουμε το ποιοι χρήστες έχουν αναφέρει μέσα στα rayz τους το token t και πόσες φορές
UserTokens[u_i][h].w, TokenUsers [u_i][h].w, UserTokens[u_i][h].w = TokenUsers [u_i][h].w	Το w ορίζεται ως ο αριθμός των φορών που έχει αναφερθεί το ενδιαφέρον h, όπου $h \in H_{p_{u_i}}$ στον χρήστη u_i (UserTokens) και αντίστοιχα ο αριθμός των φορών που το ενδιαφέρον h έχει λεχθεί από το χρήστη u_i
Send (s_j, <DATA>)	Τα δεδομένα που στέλνει ο εξυπηρετητής s_i στον εξυπηρετητή s_j
Receive (s_j, <DATA>)	Τα δεδομένα που λαμβάνει ο εξυπηρετητής s_i από τον εξυπηρετητή s_j
$O_{s_i,s_j} = T \cap Q$, $s_i \neq s_j$, $O_{s_i,s_j} \epsilon O_{s_i}$	O_{s_i,s_j} είναι τα κοινά που έχουν μεταξύ τους ο εξυπηρετητής s_i με τον εξυπηρετητή s_j
w_{u_i,r_j}, $i \neq j$	Με τον όρο $w_{u,r}$ δηλώνεται το βάρος, το πόσο «κοντά» βρίσκεται ο user u_i με το user r_j, όπου $i \neq j$, σε σχέση με τα κοινά τους ενδιαφέροντα
$dist(u_i, u_j), i \neq j$	Η Ευκλείδεια απόσταση μεταξύ του χρήστη u_i και u_j
$sim\_server_{s_i,s_j}$, $s_i \neq s_j$	Με τον όρο $sim\_server_{s_i,s_j}$ ορίζουμε το ποσοστό ομοιότητας μεταξύ των δύο εξυπηρετητών s_i και s_j για τους οποίους ισχύει $s_i \neq s_j$ και
$sim\_user_{u_i,u_j}$, $u_i \neq u_j$	Με τον όρο $sim\_user_{u_i,u_j}$ ορίζουμε το ποσοστό ομοιότητας μεταξύ των δύο χρηστών u_i και u_j για τους οποίους ισχύει $u_i \neq u_j$
$sim\_user\_server_{u_i,s_j}$	Με τον όρο $sim\_user_{u_i,s_j}$ ορίζουμε το ποσοστό ομοιότητας μεταξύ του χρήστη u_i και του εξυπηρετητή s_j
jaccard_sim (sig1, sig2)	Με την μέθοδο jaccard_sim (sig1, sig2) υπολογίζεται η ομοιότητα μεταξύ δύο υπογραφών sig1 και sig2 με βάση το Jaccard Similarity
$SIGNATURE\_SIZE$	Μέγεθος για τις υπογραφές τόσο των χρηστών όσο και των εξυπηρετητών
SIG_{u_i}	Υπογραφή – MinHash για το χρήστη u_i
SIG_{s_i}	Υπογραφή – MinHash για τον εξυπηρετητή s_i
Threshold_Server	Είναι το λιγότερο επιτρεπτό όριο ομοιότητας που πρέπει να έχουν δύο οποιοιδήποτε servers για να μπορούν να ανταλλάξουν χρήστες
Threshold_User	Είναι το λιγότερο επιτρεπτό όριο ομοιότητας που πρέπει να έχει ένας χρήστης σε σχέση με ένα server για να μπορεί να σταλεί ως μέλος του BCS προς το συγκεκριμένο server
Original_Users	Ο μετρητής Original_Users δηλώνει το πλήθος των αρχικών χρηστών για ένα εξυπηρετητή πριν να του στείλουν οι άλλοι εξυπηρετητές τους όμοιους με αυτόν χρήστες

Κεφάλαιο 4

SN-Algorithm

Σε αυτό το κεφάλαιο θα γίνει η περιγραφή των φάσεων του αλγορίθμου που υπολογίζει τους σημασιολογικά κοντινούς γείτονες για κάποιο χρήστη (SN Algorithm - Semantic Neighbor Algorithm) και θα επεξηγηθεί το τι γίνεται σε κάθε φάση εκτέλεσης του αλγορίθμου. Συνοπτικά να αναφερθεί το γεγονός ότι ο αλγόριθμος, στη γενική του μορφή, ακολουθεί μια συγκεκριμένη πορεία με συγκεκριμένα βήματα και πως οι δύο προσεγγίσεις που θα παρουσιαστούν διαφέρουν στις δομές που χρησιμοποιούν, στο τρόπο που γίνεται η ανταλλαγή των χρηστών και στη σύσταση για τους χρήστες. Γι’ αυτό στην υποενότητα 4.4

θα αναλυθούν οι κοινές φάσεις των δύο προσεγγίσεων και στις επόμενες δύο προσεγγίσεις θα αναλυθούν οι δύο διαφορετικές. Στην τελευταία υποενότητα θα γίνει μια ποιοτική σύγκριση των δύο προσεγγίσεων.

4.1 Εισαγωγή

Σε αυτή την υποενότητα θα αναφερθούν κάποιες βασικές έννοιες που δεν έχουν αναφερθεί σε προηγούμενα κεφάλαια και αφορούν κυρίως τεχνικές λεπτομέρειες που έχουν σχέση με την επίλυση αλλά και την ορθότητα των προσεγγίσεων. Επίσης θα γίνει αναφορά και σε κάποιες επιπλέον εισηγήσεις που έγιναν κατά τη διάρκεια της εφαρμογής λύσης οι οποίες και βοήθησαν περαιτέρω στην καλυτέρευση του αλγορίθμου.

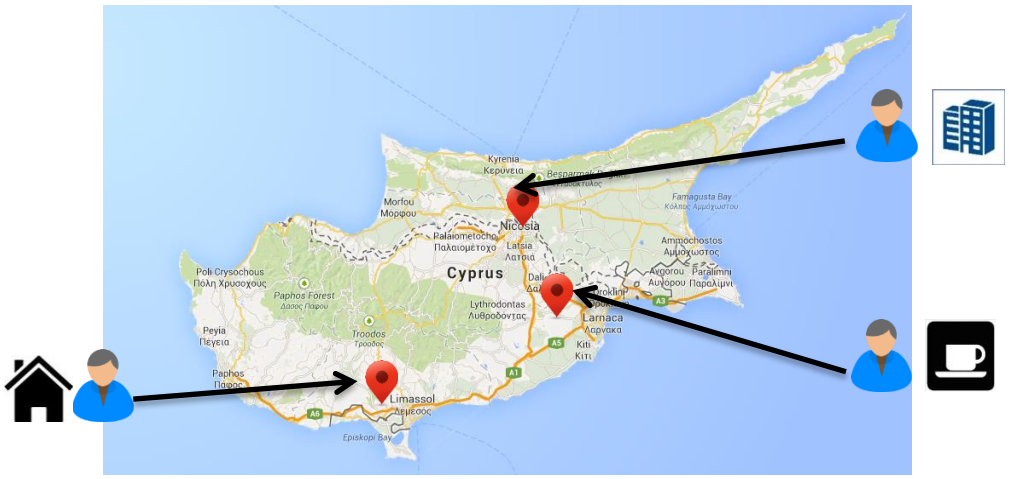
Αρχικά να αναφερθεί ότι το πρόβλημα, όπως μέχρι και τώρα, ανάγεται σε πρόβλημα τύπου online processing και όχι offline processing. Όταν επιτελείται offline processing, ουσιαστικά θα παρθεί ένα στιγμιότυπο μιας βάσης δεδομένων και θα διεξαχθεί πάνω στα δεδομένα επεξεργασία με σκοπό να εξάχθούν κάποια αποτελέσματα. Τα αποτελέσματα τις περισσότερες φορές αφορούν στατιστικά στοιχεία για διάφορους οργανισμούς, μοτίβα κτλ. Στη περίπτωση του Rayzit, ενδιαφέρει περισσότερο το online processing.

Η διαφορά του online processing σε σχέση με το offline processing βρίσκεται στο ότι οι online αλγόριθμοι κάθε φορά που κάποιος θα ζητήσει το αποτέλεσμα θα πρέπει να εκτελούνται εκ νέου. Το Online processing ορίζεται ως η εργασία που εκτελείται σε πραγματικό χρόνο και δίνει άμεσα αποτελέσματα σε αυτό που την εκτελεί σε ένα σχετικά εύλογο χρονικό διάστημα. Ενώ το offline processing πάλι αναμένεται να παρθεί αποτέλεσμα αλλά δεν είναι απαραίτητο αυτό να γίνει σε πολύ γρήγορο διάστημα λόγω του ότι τα δεδομένα είναι πολλά και ίσως αυτό να πάρει χρόνο.

Γι’ αυτό και στη περίπτωση μας, μας ενδιαφέρει σε μεγάλο βαθμό το online processing. Διότι όσο καλές και αν είναι οι προσεγγίσεις, θα πρέπει οι αλγόριθμοι που υλοποιούνται και που θα τρέχουν πάνω στη πλατφόρμα να είναι αποδοτικοί και γρήγοροι στην εκτέλεση. Ο λόγος έγκειται στο ότι η άτακτη κίνηση των χρηστών μέσα στο χώρο θα πρέπει να μας επιτρέπει να υπολογίζουμε αρκετές φορές, ακόμα και για διάστημα κάποιων λεπτών, το AkNN αλλά και το AkSNN που θα παρουσιαστεί στις επόμενες υποενότητες. Αυτό προσδίδει την αναγκαιότητα στον SNA να είναι αποδοτικός στην εκτέλεση του.

Reason: A user has different interests according to his location!
Home: Wants to learn for a hobby he has
Work: Wants to learn according to his profession
Coffee Shop: Wants to learn he amuses him

Σχήμα 4.1. Ένα χρήστης μπορεί να έχει διαφορετικά ενδιαφέροντα



Σχήμα 4.2. Ένα χρήστης μπορεί να βρίσκεται γεωγραφικά σε σχετικά κοντινά σημεία, ακόμα και στον ίδιο εξυπηρετητή, αλλά τα ενδιαφέροντα του θα δύναται να διαφέρουν από σημείο σε σημείο όταν χρησιμοποίει την υπηρεσία

Στη περίπτωση του Rayzit αυτό που θα εκτελείται κάθε ένα καθορισμένο χρονικό διάστημα είναι ο αλγόριθμος SN ο οποίος θα υπολογίζει τις συστάσεις που πρέπει να γίνουν για κάθε χρήστη. Ο αλγόριθμος είναι πολύ σημαντικό να ακολουθεί μια τέτοια προσέγγιση αφού τα δεδομένα αλλάζουν συνεχώς. Με τον όρο δεδομένα εννοούμε προφανώς τα μηνύματα των χρηστών αλλά επίσης και τη τοποθεσία τους μέσα στο κόσμο. Οι χρήστες σαν δεδομένα, documents μέσα στη document base βάση – Couchbase, δεν αλλάζουν. Δηλαδή σε κάθε στιγμιότυπο της βάσης τα documents για αντίστοιχους χρήστες δεν αλλάζουν. Αυτά που αλλάζουν είναι το περιεχόμενο τους και συγκεκριμένα η τοποθεσία τους.

Γι’ αυτό και η εκτέλεση του αλγορίθμου θα καταχωρηθεί ως μια επαναλαμβανόμενη διεργασία στο περιβάλλον UNIX (γνωστή ως cronjob) και κάθε ένα συγκεκριμένο χρονικό διάστημα το λειτουργικό θα αφυπνίζεται και θα τρέχει τον αλγόριθμο.

Σχήμα 4.3. Μπάρα βαρύτητας ποιότητος υπηρεσίας

Ένα άλλο θέμα που προέκυψε στην πορεία ήταν για το κατά πόσο θα ήταν εφικτό να γίνεται η σύσταση ανάλογα με τη γεωγραφική θέση του χρήστη. Ο χρήστης σύμφωνα με το Σχήμα 4.2, ανάλογα με το χώρο στον οποίο βρίσκεται έχει διαφορετικά ενδιαφέροντα. Στο Σχήμα 4.2 φαίνεται ότι ο συγκεκριμένος χρήστης θεάθηκε σε διαφορετικές τοποθεσίες μέσα στη Κύπρο. Σε κάθε μια από τις 3 περιπτώσεις σύμφωνα με το Σχήμα 4.1 ο χρήστης εξέφρασε διαφορετικά ενδιαφέροντα. Άρα αυτό που θέλουμε να επιτύχουμε εμείς είναι όταν θα βρίσκεται στις συγκεκριμένες τοποθεσίες να του προτείνονται χρήστες που είναι κοντά στα ενδιαφέροντα του με βάση τη τοποθεσία του.

Δηλαδή αν ο χρήστης βρίσκεται για παράδειγμα στη Λευκωσία όπου είναι ο χώρος εργασίας του θα πρέπει να φέρνει του ποιο σημασιολογικά και φυσικά κοντινούς του γείτονες οι οποίοι έχουν τις ίδιες ασχολίες και ενδιαφέροντα με αυτόν. Για παράδειγμα το android development. Ο αλγόριθμος θα φέρει τους kSNN χρήστες που βρίσκονται κοντά στη Λευκωσία και έχουν κοινό το android development. Αντίστοιχα θα γίνεται και όταν θα βρίσκεται σε άλλη πόλη.

Στο προηγούμενο κεφάλαιο έχει γίνει αναφορά για το βάρος που δίνεται για το αν θα δοθεί προτεραιότητα στη πραγματική απόσταση μεταξύ των χρηστών ή για το κατά πόσο θα δίνεται έμφαση στα κοινά ενδιαφέροντα που έχουν μεταξύ τους οι χρήστες. Η συγκεκριμένη τεχνική έχει ονομαστεί Multi-Objective και έχει επεξηγηθεί εκτενώς στο προηγούμενο κεφάλαιο. Υπάρχει η σκέψη πάνω στο πρωτότυπο σύστημα το οποίο θα παρουσιαστεί στο επόμενο κεφάλαιο, ο χρήστης να έχει τη δυνατότητα να επιλέγει και να φιλτράρει ανάλογα με το τι θέλει ο ίδιος για τα αποτελέσματα που θα παίρνει ως απάντηση όταν θα ζητά ενημέρωση από το σύστημα. Μια απλή αναπαράσταση είναι αυτή που φαίνεται στο Σχήμα 4.3 ποιο πάνω και είναι ουσιαστικά μια μπάρα κύλισης κατά την οποία όσο γεμάτη είναι προς τα δεξιά τότε θα δίνεται περισσότερη βαρύτητα στα ενδιαφέροντα μεταξύ των χρηστών και λιγότερη σε σχέση με την πραγματική τους απόσταση, και το ανάποδο στην αντίθετη περίπτωση.

Algorithm Proposal **:** SN Algorithm
Input: We take as input all the users u_i in the set U, all the rayz messages $P_{u_i} = \{ p_{u_i,1} , p_{u_i,2} , \dots , p_{u_i,f} \}$ where $\|P_{u_i}\|$ = F of each individual user u_i and we fill in the structures explained below in order to answer the AkSNN query for all users in U. **Output:** We take as an output, respectively for each u_i in the set U, the kSNN (u_i, U) which combined for all users is called AkSNN. **Replication of the algorithm to all servers and execute the following steps** 1. Partitioning Phase 2. SN Algorithm 3. Build Phase • Create Adjacency Matrix • Create RCS (R Candidate Sets) • Exchange BRSs • Build kSNN structures 4. Find Phase (Multi-Objective result for the k-SNN problem)

Πίνακας 4.4. Φάσεις του SN Algorithm

4.2 Περιγραφή Φάσεων Αλγορίθμου

Σε αυτή την υποενότητα θα μελετηθούν όλες οι φάσεις εκτέλεσης του SNA. Στον Πίνακα 4.4 παρατίθεται ο αλγόριθμος SN σε αφαιρετική μορφή. Στη συνέχεια θα επεξηγηθούν οι φάσεις του αλγορίθμου οι οποίες είναι κοινές και για τις δύο προσεγγίσεις. Πρέπει να ειπωθεί το γεγονός ότι όλα τα βήματα που θα επεξηγηθούν εκτελούνται πάνω σε όλους τους εξυπηρετητές $s_i \in S$. Υπάρχουν δύο σημεία στον αλγόριθμο όπου οι εξυπηρετητές ανταλλάζουν δεδομένα και θα επεξηγηθούν στη συνέχεια.

4.2.1 Γενικά Σχόλια για τον αλγόριθμο SN – Κατανεμημένη Προσέγγιση

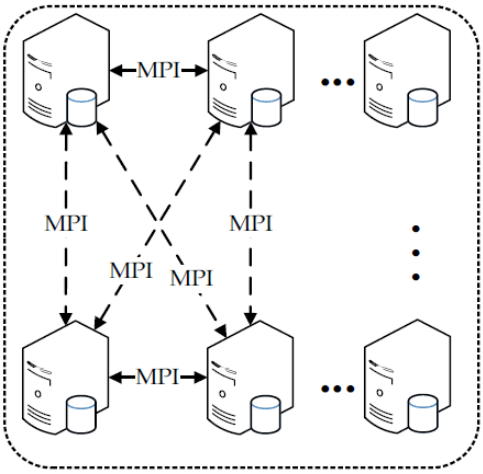
Ο αλγόριθμος SN, είναι ένας κατανεμημένος αλγόριθμος. Είναι σχεδιασμένος να δουλεύει σε ένα κατανεμημένο περιβάλλον που υποστηρίζει, οι διάφοροι υπολογισμοί να γίνονται σε διαφορετικούς επεξεργαστές και με αυτό τον τρόπο να πετυχαίνεται η αποδοτικότητα. Ο κατανεμημένος προγραμματισμός είναι ένα παρακλάδι του παράλληλου προγραμματισμού. Όπως και στον παράλληλο προγραμματισμό, έτσι και στον κατανεμημένο, ο υπολογισμός σπάζει σε διάφορα κομμάτια τα οποία θα τρέξουν ταυτόχρονα στο παράλληλο και κατανεμημένα στο κατανεμημένο προγραμματισμό.

Η βασική τους διαφορά είναι ο τρόπος που μοιράζονται και εκτελούνται τα διάφορα κομμάτια δεδομένων (chunk of data). Στο παράλληλο προγραμματισμό τα κομμάτια θα εκτελεστούν πάνω στην ίδια μηχανή. Χρησιμοποιούνται νήματα (threads) τα οποία θα αναλάβουν ένα συγκεκριμένο ποσοστό του συνολικού όγκου δεδομένων που πρέπει να τύχουν επεξεργασίας. Μπορούν να χρησιμοποιηθούν κάποιες τεχνικές όπως είναι η κοινή μνήμη όταν κάποιος επεξεργαστής που εκτελεί επεξεργασία πάνω σε ένα κομμάτι δεδομένων, θέλει να πάρει δεδομένα από κάποιο άλλο επεξεργαστή.

Συχνά αν θεωρήσουμε ότι ο συνολικός χρόνος αποπεράτωσης ενός προβλήματος με σειριακό τρόπο (κανονικό πρόγραμμα χωρίς παραλληλισμό) είναι S. Αν χρησιμοποιηθούν νήματα N, τότε ο χρόνος θα κυμανθεί κοντά στο S/N. Αυτό συμβαίνει διότι στο παραλληλισμό ο χρόνος υποδιαιρείται ανάλογα με το πόσοι επεξεργαστές έχουν αναλάβει την αποπεράτωση μιας εργασίας. Συχνά ο χρόνος δεν ακολουθεί επακριβώς το S/N αλλά το S/N + p, όπου p είναι ο χρόνος που απαιτείται για τις αρχικοποιήσεις των νημάτων, ανταλλαγής μηνυμάτων μεταξύ τους αλλά και συγχρονισμό μεταξύ τους.

Από την άλλη στον κατανεμημένο προγραμματισμό δεν συμβαίνει ακριβώς ότι συμβαίνει στον παράλληλο προγραμματισμό. Στον κατανεμημένο προγραμματισμό ένας αλγόριθμος είναι σχεδιασμένος να τρέχει πάνω σε αρχιτεκτονική που να υποστηρίζει την εκτέλεση αλγόριθμων πάνω σε κατανεμημένες υπολογιστικές μονάδες, δηλαδή ανεξάρτητες (εξυπηρετητές), οι οποίες τις περισσότερες φορές έχουν περιορισμένες πληροφορίες η μια για την άλλη, για το ποιο κομμάτι των αλγόριθμων εκτελούνται.

Ένα από τα ποιο βασικά στοιχεία που πρέπει να χαρακτηρίζουν κατανεμημένους αλγορίθμους είναι ο επιτυχής συντονισμός της συμπεριφοράς των ανεξάρτητων φάσεων ενός αλγορίθμου για την αντιμετώπιση των αδυναμιών, σε περίπτωση που μια υπολογιστική μονάδα παρουσιάσει πρόβλημα ή υπάρχει αδυναμία επικοινωνίας μεταξύ των υπολογιστικών μονάδων.



Σχήμα 4.5. Εικονική αναπαράσταση ανταλλαγής μηνυμάτων των εξυπηρετητών

Δύο βασικά χαρακτηριστικά που υπάρχουν στον κατανεμημένο προγραμματισμό είναι η αντιγραφή (replication) κατά την οποία ο αλγόριθμος αντιγράφεται στις υπόλοιπες υπολογιστικές μονάδες για να αρχίσει η κατανεμημένη του εκτέλεση και η ανταλλαγή μηνυμάτων (messaging passing interface) όπου επιτρέπει την ανταλλαγή μηνυμάτων μεταξύ των υπολογιστικών μονάδων.

4.2.2 Ανταλλαγή μηνυμάτων σε κατανεμημένα συστήματα

Το messaging passing interface που είναι γνωστό ως MPI είναι ένα καθορισμένο και φορητό σύστημα - πρωτόκολλο για μετάδοση μηνυμάτων που κατασκευάστηκε με σκοπό να βοηθήσει την ανταλλαγή μηνυμάτων που επιτελείται τόσο στον παράλληλο όσο και στον κατανεμημένο προγραμματισμό. Το συγκεκριμένο πρωτόκολλο έχει υλοποιηθεί σε αρκετές γλώσσες προγραμματισμού και μια από αυτές είναι και η JAVA. Στην περίπτωση του SNA έχει χρησιμοποιηθεί η Parallel Java όπου είναι ακριβώς ένα MPI μοντέλο όπως επεξηγήθηκε μέχρι τώρα που επιτρέπει τη μετάδοση μηνυμάτων στους εξυπηρετητές. Συνοπτικά θα δοθούν οι ποιο κάτω συναρτήσεις που επιτρέπουν την αποστολή και τη λήψη μηνυμάτων στη Parallel Java.

send

```
public void send(int toRank, Buf buffer)
```

receive

```
public CommStatus receive(Integer fromRank, Buf buffer)
```

Οι δύο αυτές συναρτήσεις επιτρέπουν την αποστολή και την λήψη μηνυμάτων από τον εξυπηρετητή toRank. Στην μεν πρώτη περίπτωση το toRank δηλώνει τον εξυπηρετητή στον οποίο θα σταλεί το μήνυμα ενώ στη δεύτερη περίπτωση τον εξυπηρετητή από τον οποίο θα

User_0 **A**

```
{
  "acc": "65.00",
  "appID": "WP8APP_AGB3PGERO054FGT3AFERA0",
  "geometry": {
    "type": "Point",
    "coordinates": [
      52.62050744140859,
      53.519688129423244
    ]
  },
  "timestamp": "1398522195336",
  "power": "100"
}
```

User_0_0 **B**

```
{
  "userId": "User_0",
  "rayz_message": "semblance's sir internes
  "geometry": {
    "type": "Point",
    "coordinates": [
      101.92820747540657,
      -25.08016900919055
    ]
  }
}
```

Σχήμα 4.6 Παράδειγμα ενός εγγράφου – document αποθήκευσης πληροφοριών για ένα
Σχήμα 4.7 Παράδειγμα ενός εγγράφου – document αποθήκευσης πληροφοριών για ένα μήνυμα rayz ενός χρήστη

λάβει το μήνυμα. Στο Σχήμα 4.5 φαίνεται πως επιτελείται η ανταλλαγή μηνυμάτων από τους εξυπηρετητές που υποστηρίζουν την υπηρεσία Rayzit. Το S/N + p που έχει αναφερθεί προηγουμένως, ισχύει και σε αυτή τη περίπτωση μόνο που το p είναι ποιο μεγάλο αφού κοστίζει αρκετά η επικοινωνία που υπάρχει μεταξύ των εξυπηρετητών. Τα μηνύματα που ανταλλάζονται σε αυτό το μοντέλο επικοινωνίας κοστίζουν περισσότερο απ' ότι τα μηνύματα ενός απλού προγράμματος που περιορίζονται τοπικά σε μια υπολογιστική μονάδα. Σε αυτή τη περίπτωση το κόστος είναι ανάλογο του δικτύου που συνδέει τους εξυπηρετητές μεταξύ τους. Όσο περισσότερο bandwidth υπάρχει τόσο ποιο λίγο είναι το κόστος.

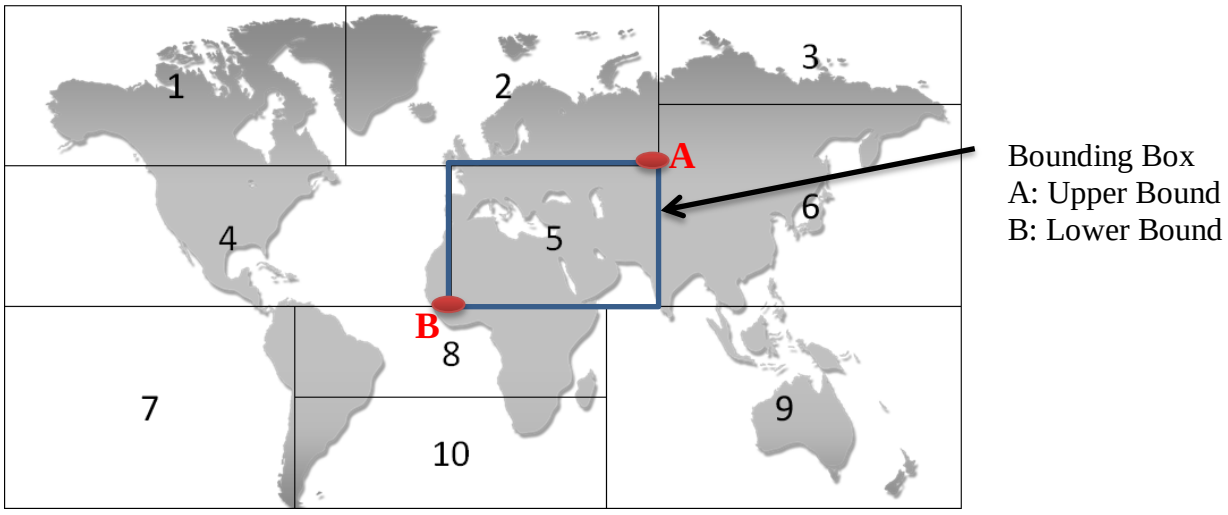
4.2.3 Βάση Δεδομένων για ημι-δομημένα δεδομένα

Σε αυτή την υποενότητα θα δοθούν κάποιες εξηγήσεις για τη βάση δεδομένων, που είναι η CouchBase, μέσα στην οποία αποθηκεύονται τα δεδομένα του Rayzit αλλά και γιατί έχει επιλεγεί η συγκεκριμένη.

Για τη συγκεκριμένη εφαρμογή έχει κριθεί απαραίτητο να χρησιμοποιηθεί η βάση CouchBase. Η βάση αυτή χειρίζεται ημι-δομημένα δεδομένα όπως είναι η JSON τύπου μορφή και τα αποθηκεύει σε μορφή εγγράφων – documents.

Στο Σχήμα 4.6 φαίνεται η δομή με την οποία αποθηκεύονται τα δεδομένα που αφορούν ένα τυχαίο χρήστη ενώ στο Σχήμα 4.7 φαίνεται η δομή με την οποία αποθηκεύονται τα δεδομένα ενός μηνύματος rayz ενός τυχαίου χρήστη.

Οι επερωτήσεις της βάσης, για την CouchBase, ονομάζονται views. Το σημαντικό για τα views είναι ότι όταν ο διαχειριστής της βάσης δημιουργήσει ένα view τότε σε πού λίγο χρονικό διάστημα δημιουργεί ευρετήριο - indexing πάνω στα δεδομένα και όταν καλεστεί το συγκεκριμένο view τότε μπορεί να δοθεί σαν είσοδος το εύρος του view και έτσι να



Σχήμα 4.8. Εικονική αναπαράσταση ενός Bounding Box που ισοδυναμεί με το καταμερισμό εργασίας εξυπηρετητή

επιστραφούν τα αποτελέσματα που ανήκουν μέσα σε αυτό το εύρος όπως φαίνεται στο Σχήμα 4.8.

Το εύρος για ένα view είναι πολύ σημαντικό γιατί όπως θα δούμε και στην επόμενη υποενότητα θα παίξει ρόλο μετά το διαχωρισμό χώρου, ποια δεδομένα θα πρέπει ο κάθε εξυπηρετητής να πάρει, σαν μονάδα επεξεργασίας, από την CoucBase. Με τα spatial views που υποστηρίζει η CouchBase θα μπορεί ο κάθε εξυπηρετητής να ασχοληθεί με συγκεκριμένους χρήστες δίνοντας σαν είσοδο το Upper και το Lower bound του Bounding Box. Έτσι το view θα επιστρέφει τους χρήστες που βρίσκονται μέσα στο εύρος που δίνεται σαν είσοδος. Με τα αποτελέσματα του διαχωρισμού χώρου (Upper και Lower Bound) θα καλείται και το view που επιστρέφει τα μηνύματα των χρηστών. Δηλαδή θα επιστρέφονται σαν αποτέλεσμα μόνο τα μηνύματα που βρίσκονται στο συγκεκριμένο εύρος.

Αφού έχει επεξηγηθεί το μοντέλο το οποίο ακολουθεί ο αλγόριθμος αλλά και η βάση δεδομένων που χρησιμοποιείται, θα πρέπει να αναφερθεί σε αυτό το σημείο η πρώτη φάση κατά την οποία γίνεται ο διαχωρισμός του χώρου και κατά συνέπεια η ισότιμη κατανομή εργασίας που επεξηγήθηκε σε προηγούμενο κεφάλαιο. Η φάση αυτή θα επεξηγηθεί στην επόμενη υποενότητα.

4.3 Διαχωρισμός Χώρου - Partitioning Phase

Η φάση κατά την οποία γίνεται ο διαχωρισμός του χώρου είναι κοινή σε όλες τις προσεγγίσεις που έχουν υλοποιηθεί για τον SNA. Ο διαχωρισμός του χώρου μπορεί να γίνει με διάφορες τεχνικές κάτι όμως που δεν θα απασχολήσει τη συγκεκριμένη Ατομική Διπλωματική Εργασία. Συνοπτικά να αναφερθεί ότι ο αλγόριθμος που έχει επιλεγεί, ανάλογα

με το πώς οι χρήστες κατανέμονται μέσα στο χώρο, που θα επιτελέσει το διαχωρισμό χώρο θα κατανέμει τους χρήστες ομοιόμορφα στους εξυπηρετητές με τρόπο που θα δώσει στον κάθε εξυπηρετητή τα όρια του (bounding box coordinates).

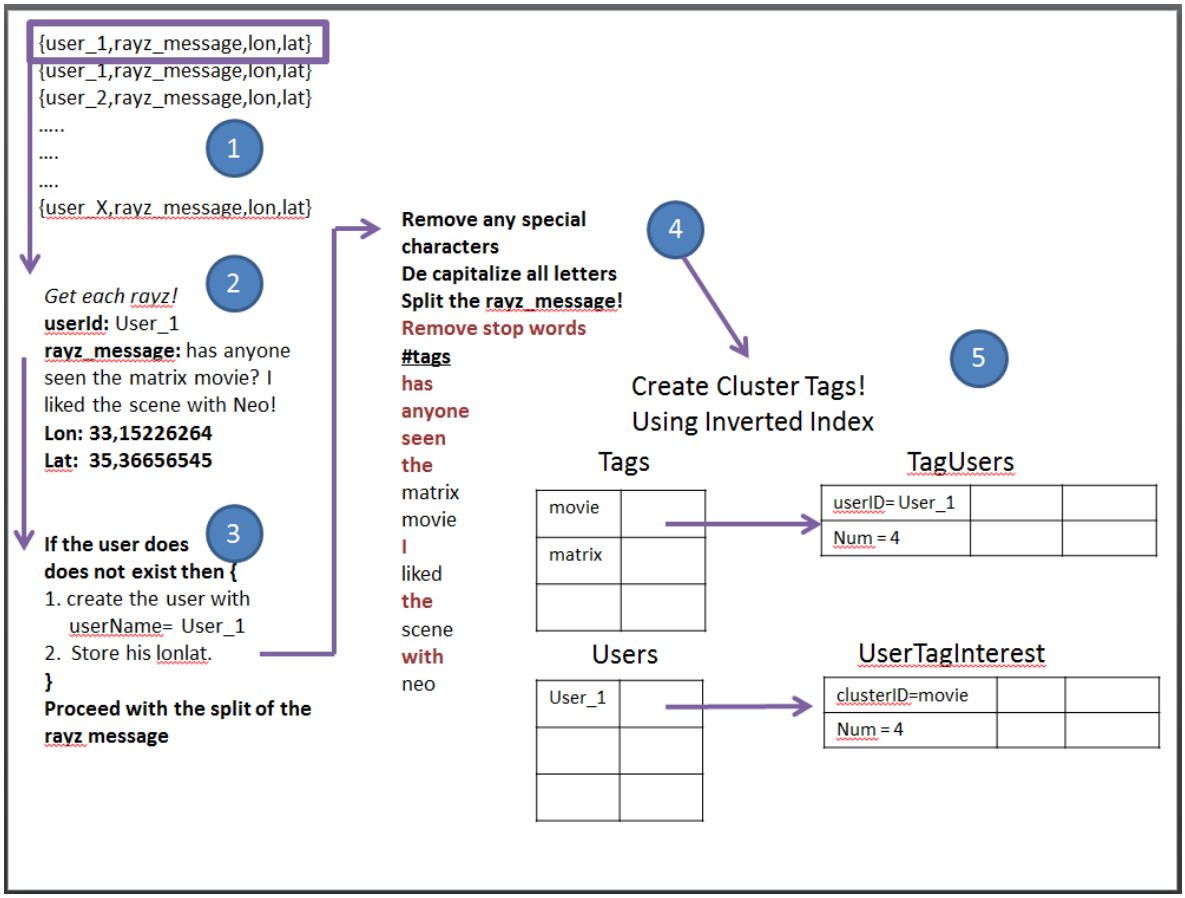
Στο Σχήμα 4.8 φαίνεται ένας πιθανός διαχωρισμός του χώρου. Ο διαχωρισμός έγινε με αυτό τον τρόπο ούτως ώστε ο κάθε εξυπηρετητής – μονάδα υπολογισμού, σύμφωνα με το κατανεμημένο υπόβαθρο που έχει επεξηγηθεί να αναλάβει να ασχοληθεί με ένα συγκεκριμένο αριθμό χρηστών, όπου ο αριθμός αυτός θα είναι ο ίδιος για όλους. Δηλαδή σύμφωνα με το σχήμα όλοι οι εξυπηρετητές θα έχουν ίδιο αριθμό χρηστών.

Αρχικά δηλαδή όλοι οι εξυπηρετητές θα πάρουν όλα τα δεδομένα που αφορούν τους χρήστες και την τοποθεσία τους. Στη συνέχεια ο αλγόριθμος για το διαχωρισμό χώρου θα βρει τα bounding box coordinates όπως αυτές φαίνονται στο Σχήμα 4.8. Ακολούθως χρησιμοποιώντας τις συντεταγμένες για τα bounding box θα καλέσει ξανά το view για τους χρήστες και θα πάρει μόνο τους χρήστες που ανήκουν στο εύρος επεξεργασίας για αυτό τον εξυπηρετητή. Το ίδιο θα συμβεί, όπως ειπώθηκε και προηγουμένως, και με τα μηνύματα των χρηστών. Χρησιμοποιώντας τις ίδιες συντεταγμένες για τα bounding box, ο κάθε εξυπηρετητής θα καλέσει το view που επιστρέφει τα μηνύματα και θα λάβει μόνο τα μηνύματα που βρίσκονται σε αυτό το εύρος για την επεξεργασία.

4.4 Αλγόριθμος SN

Σε αυτή την ενότητα θα παρουσιαστούν οι δύο προσεγγίσεις που έχουν γίνει για τη λύση του προβλήματος που έχει παρουσιαστεί στα προηγούμενα κεφάλαια. Η υλοποίηση των αλγορίθμων έχει γίνει στη γλώσσα JAVA. Στο σημείο αυτό υποθέτουμε ότι μετά τη φάση του διαχωρισμού του χώρου θα καλεστεί το spatial view της βάσης δεδομένων όπου θα δοθεί σαν είσοδος τα όρια του bounding box του εξυπηρετητή, και στο τέλος ο καθένας τους θα έχει όλα τα rayz που πρέπει όπου και θα αρχίσει η επεξεργασία.

Αρχικά πρέπει να επεξηγηθεί ο όρος token ο οποίος έχει λεχθεί αρχικά και στο κεφάλαιο του Μοντέλου Συστήματος. Το Token είναι μια οποιαδήποτε λέξη, που ένας χρήστης αναφέρει σε ένα μήνυμα του.



Πίνακας 4.9. Σχηματική Αναπαράσταση Ενημέρωσης Δομών στο Inverted Index

Παράδειγμα Token

Rayz: «Has anyone seen the Matrix movie? I liked the scene with Neo »

Από το παράδειγμα που φαίνεται στο Σχήμα 4.9 τα tokens που θα ξεχωριστούν θα είναι οι λέξεις Matrix, movie, scene, Neo και αυτό γιατί οι υπόλοιπες λέξεις είναι συνδετικές είτε ρήματα που βοηθούν το χρήστη να συντάξει το μήνυμα του και προφανώς δεν παίζουν ρόλο στα προσωπικά του ενδιαφέροντα.

Στη πρώτη λύση χρησιμοποιείται το Inverted Index[13,14,15,16] που είναι μια μέθοδος που χρησιμοποιείται σε αρκετές εφαρμογές που έχουν σχέση με τη κατεύθυνση της Πληροφορικής «Εξόρυξη Δεδομένων στο Παγκόσμιο Ιστό» και ουσιαστικά διατηρεί πληροφορίες για ένα συγκεκριμένο θέμα. Στη περίπτωση μας είναι τα tokens των μηνυμάτων rayz των χρηστών, δηλαδή τι είπε κάποιος και πόσες φορές αλλά και για ένα token ποιοι το ανέφεραν και πόσες φορές. Περισσότερα για αυτή τη προσέγγιση θα παραταθούν στην επόμενη ενότητα.

1st Approach – Inverted Index Phase 2 - SNA: Create Cluster Tags (CT)

Input: We take as input all the users u in the set U, the rayz messages U_u of each individual user and we fill in the structures explained above
Output: Inverted Index structures UserTokens and TokenUsers
Let UserTokens as explained above
Let TokenUsers as explained above
For each user u_i in U do
Let F be the total rayz messages P_{u_i} of user u_i , $u_i \in U$
For each $p_{u_i,f}$ where $i \in F = \{1 \dots . f\}$ do
Split the rayz message $p_{u_i,f}$ of the user u_i into individual chunks
Let $D_{p_{u_i}}$ be the set of all individual tokens for this rayz
Let $H_{p_{u_i}}$ be a set that its items are filtered out by the S subset that contains stop words
$T = T \cup H_{p_{u_i}}$
For each h in $H_{p_{u_i}}$ where $H_{p_{u_i}} \subseteq D_{p_{u_i}}$ do
UserTokens[u_i][h].w +=1;
TokenUsers[u_i][h].w+=1;
done
done
done

Πίνακας 4.10. Δεύτερη Φάση του Αλγορίθμου Create Cluster Tags

Στη δεύτερη λύση θέλοντας να δοκιμάσουμε μια λύση που δίνει αποτελέσματα κατά προσέγγιση, έγινε η σκέψη να υλοποιηθεί μια λύση που θα έκανε «μετάφραση» των tokens σε μια υπογραφή – signature [17,18], που θα αντικατοπτρίζει τα ενδιαφέροντα του χρήστη. Αναλυτικές λεπτομέρειες θα δοθούν στις επόμενες ενότητες.

4.5 Πρώτη Προσέγγιση – Χρήση Inverted Index

4.5.1. SN Algorithm – Επεξεργασία Δεδομένων

Στη πρώτη προσέγγιση όπως έχει λεχθεί και προηγουμένως θα γίνει η χρήση των Inverted Index. Ο αλγόριθμος, όπως φαίνεται στο Σχήμα 4.9, θα κάνει τις εξής ενέργειες. Αρχικά θα πρέπει να διαβαστούν για όλους τους χρήστες u όπου $u_i \in U_{s_i}$, όλα τα μηνύματα $P_{u_i} = \{p_{u_i,1}, p_{u_i,2}, \dots, p_{u_i,f}\}$, $|P_{u_i}| = F$ του κάθε χρήστη. Για κάθε μήνυμα θα πρέπει να πάρουμε όλα τα δυνατά tokens t που αποτελούν αυτό το μήνυμα. Ορίζεται το σύνολο T_{u_i} ως όλα τα token που ειπώθηκαν σε όλα τα μηνύματα του χρήστη. Για κάθε token t, όπου $t \in T_{u_i}$, θα πρέπει να ελέγξουμε και στη συνέχεια να απορρίψουμε όσα t δεν ανήκουν στο σύνολο S, όπου S ορίζεται ως το σύνολο των λέξεων που υπάρχουν στο αγγλικό λεξιλόγιο και που έχουν την ιδιότητα των ουσιαστικών (δεν αποτελούν άρθρα, επίθετα, αντωνυμίες, …κτλ). Στη συνέχεια για κάθε μια από αυτές τις ορθές λέξεις θα να συμπληρώσουμε τις απαραίτητες

1st Approach – Inverted Index
Phase 3.1 - SNA: Create Adjacency Matrix (AM)
Phase 3.2 – SNA : Create BCS (CBCS)
Phase 3.3 – SNA : Exchange RCS (ERCS)
Phase 3.4 – SNA : Enhance Structures (ES)

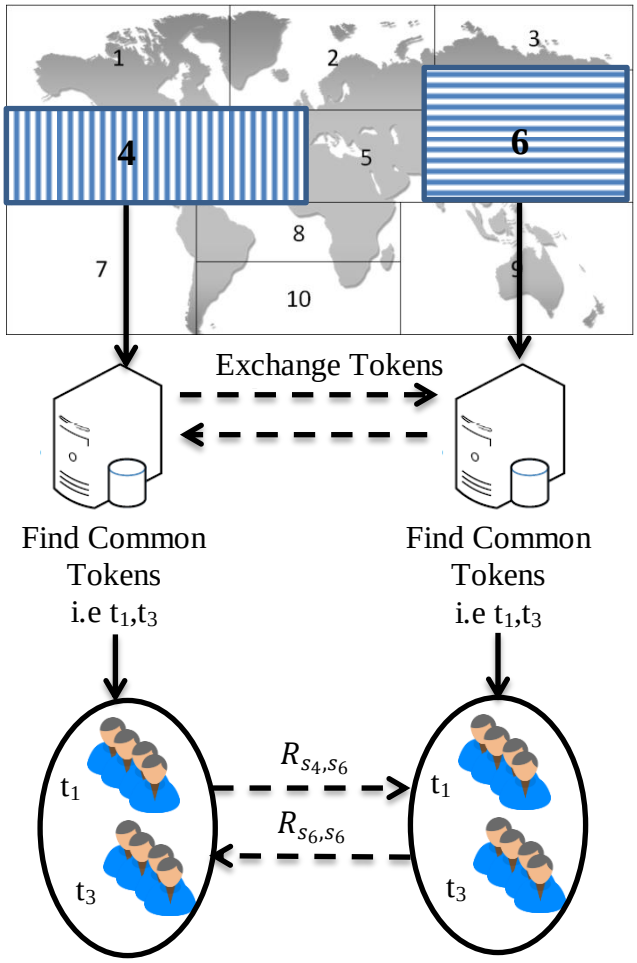
Input: Servers will exchange their tokens T
Output: At the end each server will know where to send which users according to other server's tokens T
Let T be the tokens for a specific server s_i
Let O_{s_i} be the common tokens between s_i and the rest of the servers S – $\{s_i\}$
For each server $s_j \in S$ in S do
  Send (s_j, T)
Done
For each server $\boldsymbol{s_j} \in \boldsymbol{S}$ in S do — **3.1 Create Adjacency Matrix (AM)**
  Receive (s_j, Q)
  $O_{s_i,s_j} = T \cap Q$
  For each o in O_{s_i,s_j} do — **3.2 Create RS (CRS)**
    $R_{s_i,s_j}[o] = TokenUsers[o]$
  done
done
For each R_{s_i,s_j} in R_{s_i} do — **3.3 Exchange R sets (ERS)**
    Send (s_j,R_{s_i,s_j})
done
For each R_{s_i,s_j} in R_{s_i} do — **3.4 Enhance Structures (ES)**
  Receive (s_j,R_{s_i,s_j})
  For each o in O_{s_i,s_j} do
    $TokenUsers[o] = R_{s_i,s_j}[o]$
  done
done

Πίνακας 4.12. Ανταλλαγή Tokens μεταξύ εξυπηρετητών και ενημέρωση των δομών με τους επιπρόσθετους χρήστες

δομές που θα μας βοηθήσουν να κάνουμε το σύσταση στη τελευταία φάση του αλγορίθμου. Θα χρησιμοποιηθούν δύο κύριες δομές όπως αυτές φαίνονται στον Πίνακα 4.10.

4.5.2. SN Algorithm – Επεξήγηση Αλγορίθμου

Η μια δομή είναι η TokenUsers και η άλλη UserTokens, όπου αντίστοιχα για τη πρώτη δομή, για κάθε token t, όπου t ∈ T_{u_i}, φυλάσσουμε το ποιοι χρήστες έχουν αναφέρει μέσα στα rayz τους το token t και πόσες φορές, ενώ αντίστοιχα στην άλλη δομή φυλάσσουμε για κάθε χρήστη u_i ποια tokens έχει αναφέρει στα μηνύματα του και πόσες φορές. Το σύνολο T περιέχει όλα τα πιθανά token που έχουν βρεθεί μέσα στα μηνύματα. Ισχύει ότι | T | = | TokenUsers | . Αξιοσημείωτο να αναφερθεί η σχέση TokenUsers[t] u_i] = UserTokens[u_i][t]. Η σχέση αυτή δηλώνει το πόσες φορές ο χρήστης u_i έχει αναφέρει το token t μέσα στα rayz



Σχήμα 4.12. Ανταλλαγή Tokens μεταξύ εξυπηρετητών και αποστολή χρηστών με κοινά ενδιαφέροντα

του. Ο αλγόριθμος φαίνεται σε ψευδοκώδικα στο Πίνακα 4. αλλά και με σχηματική αναπαράσταση στο Σχήμα 4.12

Στη συνέχεια θα πρέπει οι εξυπηρετητές να ανταλλάξουν όλα τα tokens που παρουσιάστηκαν στους χρήστες τους οποίους έχουν επεξεργαστεί και να δημιουργηθούν τα R σύνολα τα οποία θα περιέχουν τους χρήστες που πρέπει να ανταλλαγούν στους υπόλοιπους εξυπηρετητές. Αν για παράδειγμα οι εξυπηρετητές 4 και 6,όπως φαίνεται και σχηματικά στο Σχήμα 4.12, έχουν κοινά token t_1 και t_3, τότε το R_{s_4,s_6} και $R_{s_6,4}$θα περιέχουν στη πρώτη περίπτωση τους χρήστες του κάθε εξυπηρετητή που τους ενδιαφέρει το t_1 και t_3 και αντίστοιχα και στη δεύτερη περίπτωση θα συμβεί το ίδιο από τη πλευρά του εξυπηρετητή 6. Ακολούθως θα γίνει ενημέρωση των υπαρχουσών δομών δεδομένων που υπάρχουν σε κάθε εξυπηρετητή με τους επιπρόσθετους χρήστες που ο κάθε εξυπηρετητής θα στείλει σε κάποιον άλλο.

1st Approach – Inverted Index
Phase 4 - SNA : Find Phase (FF)
Recommendation Algorithm

Input: We take as input the Inverted Index structures

Output: We produce the recommendations per user. We actually perform User-User Recommendation and for each user we recommend other users that are similar-close to each user

For each user u_i , $u_i \in U$ in UserTokens do
  Take all u_i interests T_{u_i}
  Let K_{u_i} be a recommendation list for user u_i
  For each t in T_{u_i} do
    Get all users r_i, where $r_i = U - \{u_i\}$ that mentioned t from structure TokenUsers[t].
    Let this set be denoted as R
    For each r_i in R do
      $K_{u_i}[r_i].w_{u_i,r_j} = w_{u_i,r_j} +$ UserTokens $[u_i][\ t\ ]$ x TokenUsers $[\ t\ ][\ r_i\ ]$
     Calculate $d = \text{dist}(u_i, r_j), i \neq j$
      $K_{u_i}[r_i].d = d$
    done
  done
  For each r_i in R do
    /* normalize weight*/
    /* normalize distance*/
     $K_{u_i}[r_i].final\_value = 0.5 * K_{u_i}[r_i].d + 0.5 * K_{u_i}[r_i].w_{u_i,r_j}$
  Done
  sort with ascending order set K_{u_i} according to final value (
done

Σχήμα 4.13. Αλγόριθμος Σύστασης για τη πρώτη προσέγγιση

Σε αυτό το σημείο αφού έχουν ανταλλαγεί επιτυχώς οι χρήστες μεταξύ των εξυπηρετητών θα τρέξει ο αλγόριθμος σύστασης που χρησιμοποιεί τη δομή UserTokens. Όπου για το user u_i θα πάρουμε όλα τα token T_{u_i} και μετά για κάθε token t, όπου ισχύει ότι $t \in T_{u_i}$, χρησιμοποιώντας το inverted index TokenUsers να πάρουμε όλους τους user που ανέφεραν τα συγκεκριμένα tokens. Όμως για κάθε user r_i, όπου $r_i = U - \{u_i\}$ προσπαθώντας να το φέρουμε ποιο κοντά στον user u_i θα χρησιμοποιήσουμε την τιμή που έχει στο UserTokens $[u_i][\ t\ ]$ όπου θα την πολλαπλασιάζουμε με κάθε τιμή που βρίσκουμε στο TokenUsers $[\ t\ ][\ r_i\ ]$. Άρα θα ορίζουμε ως βάρος μεταξύ του χρήστη u_i και r_i, w_{u_i,r_j}= UserTokens $[u_i][\ t\ ]$ x TokenUsers $[\ t\ ][\ r_i\ ]$. Θα πρέπει να πούμε ότι αρχικά $w_{u_i,r_j} = 0$. Και πως στη συνέχεια καθώς τρέχει ο αλγόριθμος το w_{u_i,r_j} θα ορίζεται ως $w_{u_i,r_j} = w_{u_i,r_j}$ + UserTokens $[u_i][\ t\ ]$ x TokenUsers $[\ t\ ][\ r_i\ ]$.

Στο τέλος αφού θα έχουμε συγκεντρώσει όλους τους πιθανούς χρήστες – συστάσεις, για το χρήστη u_i θα περάσουμε πάνω από όλους και θα υπολογίσουμε την ευκλείδεια απόσταση μεταξύ των χρηστών και στη συνέχεια θα αφαιρέσουμε το βάρος που έχουμε βρει στο

προηγούμενο βήμα. Αυτό το βάρος θα φέρει ποιο κοντά τους δύο χρήστες εκμηδενίζοντας την μεγάλη νοητή-πραγματική απόσταση που είχαν μόνο με την ευκλείδεια απόσταση αφού τώρα θα βρίσκονται ποιο κοντά αφού έχουν κοινά ενδιαφέροντα. Οι ευκλείδειες αποστάσεις και τα διαφορετικά βάρη που έχουν μεταξύ τους οι χρήστες γίνονται normalize έτσι ώστε να μην ευνοείται κάποιος χρήστης περισσότερο ως προς την απόσταση ή το βάρος. Στη συνέχεια θα δοθεί 50%.

4.5.3. SN Algorithm – Πολυπλοκότητα

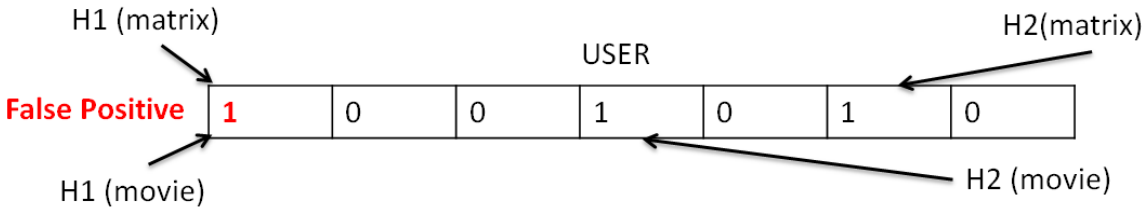
Η πρώτη προσέγγιση χωρίζεται σε 3 φάσεις όπως έχει επεξηγηθεί και ποιο πάνω. Στη πρώτη φάση γίνεται η επεξεργασία δεδομένων. Το κόστος για την επεξεργασία ενός rayz ισούται με *processing_cost* το οποίο είναι σταθερό και ισχύει και στις δύο προσεγγίσεις που θα δούμε στη συνέχεια. Επίσης στη πρώτη φάση υπάρχει το κόστος για την δημιουργία των δομών που στη περίπτωση μας επειδή τα Inverted Index χρησιμοποιούν HashMap της JAVA, αυτό μας δίνει το πλεονέκτημα της γρήγορης εισαγωγής και αναζήτησης πάνω σε αυτό. Η χρήση των HashMap γίνεται και στη δεύτερη προσέγγιση. Θέτουμε ως κόστος επεξεργασίας το processing cost – PC.

$$PC = O\left(U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser * H_{p_{u_i}} * SearchToken\right)$$

Η πολυπλοκότητα για την επεξεργασία προκύπτει όπως φαίνεται ποιο πάνω. Για U_{s_i} που είναι οι χρήστες ενός εξυπηρετητή, έχουμε ένα σταθερό κόστος για την επεξεργασία όλων των μηνυμάτων τους επί το κόστος αναζήτησης στο HashMap για να βρούμε το χρήστη όπου για κάθε token χρειαζόμαστε να αναζητήσουμε μέσα σε δεύτερο HashMap για να αναζητήσουμε και να ενημερώσουμε τα βάρη. Πρακτικά επειδή η αναζήτηση στα HashMap είναι O(1) τότε καταλήγουμε στην πολυπλοκότητα που φαίνεται ποιο κάτω.

$$PC = O\left(U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser * H_{p_{u_i}} * SearchToken\right)$$

Στη συνέχεια θα πρέπει να υπολογίσουμε το κόστος επικοινωνίας μεταξύ των εξυπηρετητών. Θα υπολογιστεί η τομή των token που έχει ένας εξυπηρετητής και θα αποσταλούν οι κατάλληλοι χρήστες ανάλογα με το πόσοι χρήστες βρίσκονται πάνω σε κάθε token. Το κόστος μεταφοράς ενός χρήστη είναι σταθερό και θα αποτιμηθεί στα πειράματα στο επόμενο κεφάλαιο. Θέτουμε ως κόστος επικοινωνίας το communication cost – CC.



Σχήμα 4.14. Παράδειγμα Υπογραφής για ένα τυχαίο χρήστη σε σχέση με τα ενδιαφέροντα του

$$R_{s_i,s_j} \cap R_{s_j,s_i} = Common\_Tokens$$

$$CC = O\left(\sum_{i=1}^{m} Common_{Tokens} * Users_{per_{token}} * Cost_{per_{user}}\right)$$

Τέλος θα πρέπει να αξιολογηθεί το κόστος πολυπλοκότητας για τον αλγόριθμο σύστασης. Η πολυπλοκότητα προκύπτει από το ποιο κάτω τύπο. Θέτουμε ως κόστος σύστασης το recommendation cost – RC.

$$RC = O\left(U_{s_i} * \boldsymbol{T_{u_i}} * Users_{per_{token}}\right)$$

Οι τελική πολυπλοκότητα για αυτή τη προσέγγιση συνοψίζεται στο επόμενο τύπο.

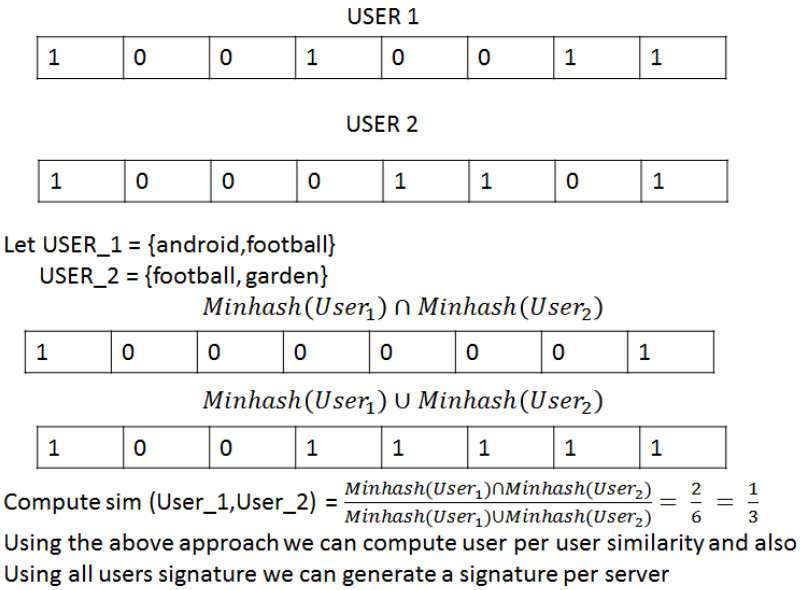
$$O\left(U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser * H_{p_{u_i}} * SearchToken + \sum_{i=1}^{m} Common_{Tokens} * Users_{per_{token}} * Cost_{per_{user}} + U_{s_i} * \boldsymbol{T_{u_i}} * Users_{per_{token}}\right)$$

Η πλήρης αξιολόγηση της πολυπλοκότητας μπορεί να γίνει μέσω των πειραμάτων και θα επεξηγηθεί στο επόμενο κεφάλαιο.

4.6 Δεύτερη Προσέγγιση – Χρήση Υπογραφών – Signatures

Η δεύτερη προσέγγιση όπως έχει επεξηγηθεί και προηγουμένως, βασίζεται στη χρήση υπογραφών – signatures. Η κύρια ιδέα είναι να μεταφραστούν τα ενδιαφέροντα των χρηστών σε υπογραφές. Για να γίνει καλύτερα κατανοητό το τι υπονοείται με τον όρο υπογραφές παρατίθεται ένα παράδειγμα στο Σχήμα 4.14.

Στην ουσία για το κάθε χρήστη $\boldsymbol{u_i}$ όπου $\boldsymbol{u_i} \in \boldsymbol{U_{s_i}}$ και για κάθε μήνυμα $P_{u_i} = \{ p_{u_i,1}, p_{u_i,2}, \dots, p_{u_i,f} \}$, $|P_{u_i}| = F$ και για το κάθε token t, όπου ισχύει $t \in p_{u_i,f}$ και $t \in D_{p_{u_i}}$ θα παρθούν όλα τα token που αντικατοπτρίζουν τα ενδιαφέροντα του χρήστη u_i, για



Σχήμα 4.15. Τρόπος υπολογισμού ομοιότητας μεταξύ δύο χρηστών

το μήνυμα $p_{u_i,f}$, που είναι $H_{p_{u_i}}$, $h \in H_{p_{u_i}}$. Το σύνολο $H_{p_{u_i}}$ περιέχει όλες οι λέξεις που αντικατοπτρίζουν τα ενδιαφέροντα που προκύπτουν στο μήνυμα $p_{u_i,f}$. Συνολικά όλα τα $H_{p_{u_i}}$ συνθέτουν το $\boldsymbol{T_{u_i}}$ που είναι τα συνολικά ενδιαφέροντα για το χρήστη $\boldsymbol{u_i}$ όπως αυτά προκύπτουν από τα μηνύματα που έχει υποβάλει στην υπηρεσία.

Ο αλγόριθμος στη συνέχεια το κάθε ένα από αυτά τα tokens t, όπου $t \in \boldsymbol{T_{u_i}}$, θα εφαρμόσει συνάρτηση κατακερματισμού στη λέξη αυτή και αναλόγως του ακεραίου αριθμού που θα προκύψει θα εφαρμοστεί μόδιο (MOD) με τον συνολικό αριθμό των θέσεων του BitSet και από 0 που είναι το bit αρχικά θα το κάνει 1. Η υπογραφή για την οποία γίνεται λόγος είναι αυτή η διαδικασία. Αρχικά το BitSet είναι αρχικοποιημένο με 0 και κάθε νέα λέξη που έρχεται περνά από αρκετές συναρτήσεις κατακερματισμού και το αποτέλεσμα κάθε συνάρτησης θα γυρίσει – flip μια θέση πάνω στην υπογραφή.

4.6.1. Κύρια Ιδέα BloomFilter

Για τις συναρτήσεις κατακερματισμού έχουν χρησιμοποιηθεί η MurmurHash και η FNVHash οι οποίες και χρησιμοποιούνται και στο BloomFilter. Το BloomFilter είναι μια αποδοτική δομή που χρησιμοποιείται σε περιπτώσεις για να ελεγχθεί αν ένα στοιχείο βρίσκεται η όχι σε ένα σύνολο δεδομένων. Το πώς δουλεύει είναι με το εξής παράδειγμα. Σύμφωνα με το Σχήμα 4.15 έστω ότι χρησιμοποιούνται οι δύο συναρτήσεις κατακερματισμού που προαναφέρθηκαν και είναι η H1 και H2 αντίστοιχα. Έστω ότι έχουμε τη λέξη «car».

H1(car) = 7 MOD SIZE_SIGNATURE (=7) = 0
H1(car) = 9 MOD SIZE_SIGNATURE (=7) = 2

2st Approach – MinHashing Approach Phase 2 – SNA : Create Signatures (CS)

Input: We take as input all the users u in the set U, the rayz messages U_u of each individual user and we fill in the structures explained above
Output: Signatures SIG_{u_i} for each individual user u_i and at the end the server's signature SIG_{s_i}
For each user u_i in U do
Let F be the total rayz messages P_{u_i} of user u_i , $u_i \in U$
Construct SIG_{u_i} and initialize with zero value
For each $p_{u_i,f}$ where i ϵ F = {1 … . f} do
Split the rayz message $p_{u_i,f}$ of the user u_i into individual chunks
Let $D_{p_{u_i}}$ be the set of all individual tokens for this rayz
Let $H_{p_{u_i}}$ be a set that its items are filtered out by the S subset that contains stop words
$T = T \cup H_{p_{u_i}}$
For each h in $H_{p_{u_i}}$ where $H_{p_{u_i}} \subseteq D_{p_{u_i}}$ do
SIG_{u_i} [MurmurHash ($H_{p_{u_i}}$) % SIGNATURE_SIZE] = 1
SIG_{u_i} [FNVHash ($H_{p_{u_i}}$) % SIGNATURE_SIZE] = 1
done
done
done
Construct SIG_{s_i} and initialize with zero value
For each user u_i in U do
$SIG_{s_i} = SIG_{s_i} \vee SIG_{u_i}$
done

Πίνακας 4.16. Δεύτερη Φάση του Αλγορίθμου για τη 2η Προσέγγιση Create Signatures

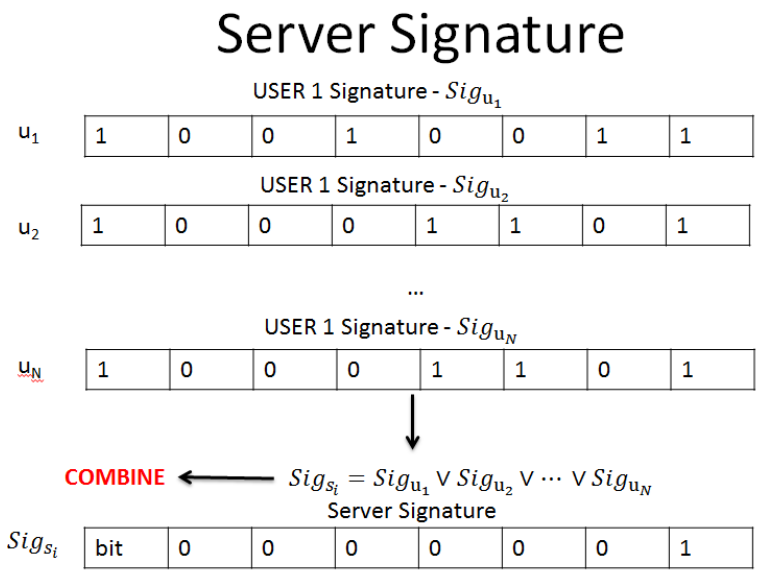
Από το ποιο πάνω παράδειγμα βλέπουμε ότι η λέξη car δεν ανήκει στο σύνολο δεδομένων που δημιούργησε την υπογραφή. Αρχικά η H1 έδωσε την θέση 0 όπου υπάρχει 1. Άρα πιθανολογικά η λέξη αυτή υπήρχε στο σύνολο δεδομένων που δημιούργησε την υπογραφή.

Όμως επειδή στο H2 η θέση που έδωσε είναι η 2 και στη 2 δεν υπάρχει 1 τότε η λέξη αυτή ποτέ δεν υπήρχε μέσα στο σύνολο που δημιούργησε την υπογραφή.

Ένα παράδειγμα που καταδεικνύει το πρόβλημα αυτής της προσέγγισης είναι στην εξής περίπτωση

H1(football) = 7 MOD SIZE_SIGNATURE (=7) = 0

H1(football) = 10 MOD SIZE_SIGNATURE (=7) = 3



Σχήμα 4.17. Δημιουργία υπογραφής εξυπηρετητή

Η λέξη football δεν υπάρχει στα δεδομένα εντούτοις ο τρόπος που δουλεύει ο συγκεκριμένος αλγόριθμος υπάρχει η περίπτωση να επιστρέψει αποτέλεσμα ότι υπάρχει. Αυτό που πρέπει να τύχει της προσοχής είναι το πόσο είναι το μέγεθος της υπογραφής που θέτουμε. Το πόσο είναι το μέγεθος θα μελετηθεί και πειραματικά.

4.6.2. SN Algorithm – Επεξήγηση Αλγορίθμου

Αφού έχει επεξηγηθεί ο τρόπος με τον οποίο εξάγεται η υπογραφή για τα ενδιαφέροντα ενός χρήστη θα πρέπει στο σημείο αυτό να επεξηγηθεί ο τρόπος με τον οποίο υπολογίζεται πώς δύο χρήστες ή/και εξυπηρετητές είναι όμοιοι και ποιο είναι το όριο που θέτουμε για να ορίσουμε στο τέλος αν είναι όμοιοι ή όχι. Ο τρόπος υπολογισμού φαίνεται στο Σχήμα 4.15.

Στην ουσία θα χρησιμοποιηθεί το Jaccard Similarity. Το Jaccard Similarity, ορίζεται από τη βιβλιογραφία, ως το πηλίκο της τομής δύο συνόλων διά την ένωση τους. Αυτό θα μας δώσει το ποσοστό ομοιότητας για δύο χρήστες και για δύο εξυπηρετητές. Για να δημιουργηθεί η υπογραφή ενός εξυπηρετητή θα ακολουθηθεί αυτό που φαίνεται στο Σχήμα 4.17. Θα γίνει σύμπτυξη – combine όλων των υπογραφών των χρηστών ενός εξυπηρετητή σε μια που θα αντικατοπτρίζει τα ενδιαφέροντα όλου του εξυπηρετητή. Η ψηφιακή υπογραφή που αντικατοπτρίζει όλα τα ενδιαφέροντα που υπάρχουν σε ένα εξυπηρετητή ορίζεται ως SIG_{S_i}. Στο Πίνακα 4.16 παρατίθεται ο αλγόριθμος που έχει επεξηγηθεί μέχρι τώρα σε αφαιρετική μορφή. Στο τέλος θα είναι έτοιμες οι υπογραφές για τους χρήστες και για τον κάθε εξυπηρετητή.

Στη συνέχεια το επόμενο βήμα θα είναι να ανταλλαγούν αυτές οι υπογραφές μεταξύ των εξυπηρετητών και ο κάθε εξυπηρετητής να συγκρίνει την ομοιότητα που έχει σε σχέση με τις

2st Approach – MinHashing Signatures Phase 3.1 – SNA : Create Adjacency Matrix (AM) Phase 3.2 - SNA : Create RCS (CRCS) Phase 3.3 - SNA : Exchange RCS (ERCS) Phase 3.4 – SNA : Enhance Structures (ES)	
Input: Servers will exchange their signatures SIG_{s_i} **Output:** At the end each server will know where to send which users according to other servers signatures	
For each server $s_j \in S$ in S do Send (s_j, SIG_{s_i}) Done	
For each server $\boldsymbol{s_j} \in \boldsymbol{S}$ in S do Receive (s_j, SIG_{s_j})	**3.1 Create Adjacency Matrix (AM)**
/*Calculate Jaccard Similarity between s_i , s_j*/ $sim\_server_{s_i,s_j}$ = jaccard_sim (s_i , s_j)	

If ($sim\_server_{s_i,s_j}$ > Threshold_Server) then
For each user u_i in U do
/*Calculate Jaccard Similarity between u_i, s_j*/
$sim\_user\_server_{u_i,s_j}$ = jaccard_sim (u_i , s_j)
If ($sim\_user\_server_{u_i,s_j}$ > Threshold_User) then
RCS_{s_i,s_j} .add (u_i)
fi
done
fi
	3.2 Create BCS (CRCS)

For each RCS_{s_i,s_j} in RCS_{s_i} do Send (s_j,RCS_{s_i,s_j}) done	**3.3 Exchange RCS (ERCS)**
Original_Users = | U | For each RCS_{s_i,s_j} in RCS_{s_i} do Receive (s_j,RCS_{s_i,s_j}) U = U ∪ RCS_{s_i,s_j} done	**3.4 Enhance Structures (ES)**

Πίνακας 4.18. Τρίτη Φάση του Αλγορίθμου για τη 2η Προσέγγιση

υπογραφές των άλλων εξυπηρετητών που έχει λάβει. Αν η ομοιότητα είναι πέραν κάποιου ορίου threshold, που ονομάζεται $sim\_server_{s_i,s_j}$, $s_i \neq s_j$, τότε οι δύο εξυπηρετητές θα ανταλλάξουν χρήστες.

Οι χρήστες όμως που θα ανταλλαγούν δεν θα είναι όλοι. Αλλά μόνο όσοι έχουν ομοιότητα πάνω από ένα συγκεκριμένο όριο με τον κάθε εξυπηρετητή με τον οποίο ελέγχονται. Αν η ομοιότητα – similarity μεταξύ ενός χρήστη και ενός εξυπηρετητή είναι πέραν κάποιου ορίου threshold, που ονομάζεται *threshold_user*, τότε ο χρήστης αυτό θα σταλεί στο συγκεκριμένο

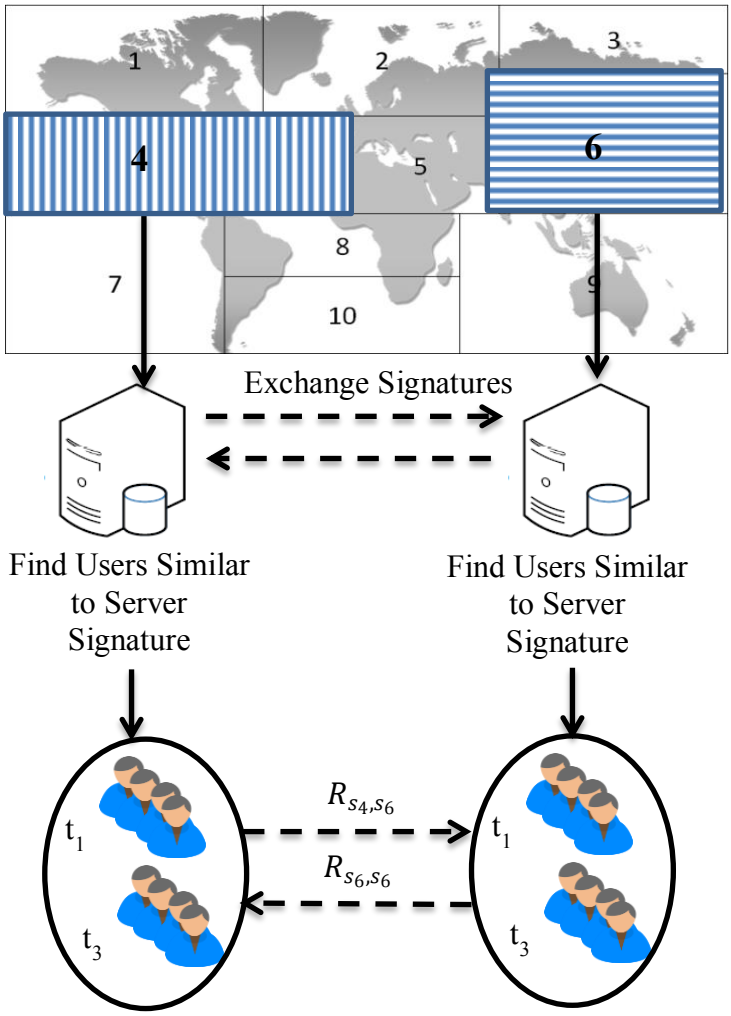
2nd Approach – MinHashing Signatures Phase 4 - SNA Algorithm : Find Phase (FF) Recommendation Algorithm

Input: We take as input the total user that exist after the enhance of the structures
Output: We produce the recommendations per user. We actually perform User-User Recommendation and for each user we recommend other users that are similar-close to each user
For the first *Original_Users* u_i , $u_i \in U$ do
For all the users u_i , $u_i \in U$ do
 	
/*Calculate Jaccard Similarity between u_i , u_j*/
$sim\_user_{u_i,u_j}$ = jaccard_sim (u_i , u_j)
 	
if ($sim\_user_{u_i,u_j}$ > Threshold_User) then
Calculate d = dist $(u_i , u_j), i \neq j$
/*subtract from the Euclidean distance the weight. Bring closer the user*/
$K_{u_i}[u_j].w_{u_i,u_j}$ = d - $sim\_user_{u_i,u_j}$
fi
done
sort with ascending order set K_{u_i}
done

Πίνακας 4.19. Αλγόριθμος σύστασης για τη 2η Προσέγγιση

εξυπηρετητή. Στο Πίνακα 4.18 παρατίθεται ο αλγόριθμος που επιτελεί την ανταλλαγή των υπογραφών μεταξύ των εξυπηρετητών καθώς και ο αλγόριθμος με τον οποίο θα γίνει η ανταλλαγή των χρηστών μεταξύ των εξυπηρετητών. Επίσης παρατίθεται ο αλγόριθμος και σχηματικά στο Σχήμα 4.20.

Στη συνέχεια θα πρέπει να τρέξει ο αλγόριθμος που θα υπολογίσει τις συστάσεις για το κάθε χρήστη όπως φαίνεται στο Πίνακα 4.19. Στο τέλος αφού θα έχουμε συγκεντρώσει όλους τους πιθανούς χρήστες – συστάσεις, για το χρήστη u_i θα περάσουμε πάνω από όλους και θα υπολογίσουμε την ευκλείδεια απόσταση μεταξύ των χρηστών και στη συνέχεια θα αφαιρέσουμε το βάρος που έχουμε βρει στο προηγούμενο βήμα και είναι η ομοιότητα - similarity. Αυτό το βάρος θα φέρει ποιο κοντά τους δύο χρήστες εκμηδενίζοντας την μεγάλη νοητή-πραγματική απόσταση που είχαν μόνο με την ευκλείδεια απόσταση αφού τώρα θα βρίσκονται ποιο κοντά αφού έχουν κοινά ενδιαφέροντα. Και σε αυτή τη προσέγγιση οι ομοιότητες και τα οι αποστάσεις γίνονται normalize.



Σχήμα 4.20. Ανταλλαγή Υπογραφών μεταξύ εξυπηρετητών και αποστολή χρηστών με κοινά ενδιαφέροντα

4.6.3. SN Algorithm – Πολυπλοκότητα

Η δεύτερη προσέγγιση χωρίζεται σε 3 φάσεις όπως έχει επεξηγηθεί και ποιο πάνω. Στη πρώτη φάση γίνεται η επεξεργασία δεδομένων. Το κόστος για την επεξεργασία ενός rayz ισούται με *processing_cost* το οποίο είναι σταθερό και ισχύει όπως και προηγουμένως. Γίνεται και εδώ η χρήση των HashMap της JAVA, αυτό μας δίνει το πλεονέκτημα της γρήγορης εισαγωγής και αναζήτησης πάνω σε αυτό.

$$PC = O\left(U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser * H_{p_{u_i}} * Hashing_{token}\right)$$

Η πολυπλοκότητα για την επεξεργασία προκύπτει όπως φαίνεται ποιο πάνω. Για U_{s_i} που είναι οι χρήστες ενός εξυπηρετητή, έχουμε ένα σταθερό κόστος για την επεξεργασία όλων των μηνυμάτων τους επί το κόστος των K συναρτήσεων κατακερματισμού πάνω στα token.

Στη συνέχεια θα πρέπει να υπολογίσουμε το κόστος επικοινωνίας μεταξύ των εξυπηρετητών. Θα υπολογιστεί η ομοιότητα που έχει ένας εξυπηρετητής και θα αποσταλούν οι κατάλληλοι χρήστες στους εξυπηρετητές. Το κόστος μεταφοράς ενός χρήστη είναι σταθερό και θα αποτιμηθεί στα πειράματα στο επόμενο κεφάλαιο. Αρχικά θα βρούμε την ομοιότητα που έχουν δύο εξυπηρετητές

$$sim\_server_{s_i,s_j}, \qquad s_i \neq s_j$$

$$R_{s_i,s_j} = users\ from\ i\ to\ j\ which\ have\ sim\_user\_server_{u_i,s_j} > threshold\_user$$

$$CC = O\left(\sum_{j=1}^{m} R_{s_i,s_j}\right)$$

Στην ανταλλαγή ισχύει ότι η πολυπλοκότητα θα προκύψει από το πόσοι χρήστες σύμφωνα με το Jaccard Similarity θα είναι όμοιοι αρκετά ώστε να ανταλλαγούν. Άρα το R_{s_i,s_j} θα αποστείλει από τον εξυπηρετητή i στον j όσους χρήστες ικανοποιούν αυτή τη συνθήκη. Η πολυπλοκότητα για τις συστάσεις προκύπτει από τον εξής τύπο όπως φαίνεται ποιο κάτω.

Οι τελική πολυπλοκότητα για αυτή τη προσέγγιση συνοψίζεται στο επόμενο τύπο. Ουσιαστικά επειδή το μόνο στοιχείο που έχουμε για τους χρήστες είναι μόνο η υπογραφή θα πρέπει ο αλγόριθμος να τρέξει για όλους τους χρήστες του εξυπηρετητή, άρα για U_{s_i} και θα πρέπει να ελέγξει με όλους τους χρήστες που υπάρχουν ήδη στην εξυπηρετητή και επιπρόσθετα για όλους όσους έχουν έρθει από άλλους εξυπηρετητές.

$$RC = O(U_{s_i} * (U_{s_i} + \sum_{j=1}^{m} R_{s_j,s_i}))$$

Σε αυτό το σημείο βρίσκεται και το πρόβλημα σε αυτή τη προσέγγιση το οποίο και δημιουργεί προβλήματα τα οποία θα επεξηγηθούν στην επόμενη υποενότητα. Η πλήρης αξιολόγηση της πολυπλοκότητας μπορεί να γίνει μέσω των πειραμάτων και θα επεξηγηθεί στο επόμενο κεφάλαιο.

Πολυπλοκότητα για δεύτερη προσέγγιση

$$O\left(U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser * H_{p_{u_i}} * Hashing_{token} + \sum_{j=1}^{m} R_{s_i,s_j} + U_{s_i} * \left(U_{s_i} + \sum_{j=1}^{m} R_{s_j,s_i}\right)\right) ,$$

$$R_{s_i,s_j} = users\ from\ i\ to\ j\ which\ have\ sim\_user\_server_{u_i,s_j} > threshold\_user,$$

$$sim\_server_{s_i,s_j}, \quad s_i \neq s_j$$

4.7 Ποιοτική Σύγκριση των δυο προσεγγίσεων

Οι δυο προσεγγίσεις που έχουν υλοποιηθεί λύνουν το πρόβλημα με τις συστάσεις το οποίο έχει προταθεί για τη πλατφόρμα Rayzit.

Η δεύτερη όμως προσέγγιση έχει ένα ελάττωμα το οποίο θα πρέπει να προβληματίσει. Στη δεύτερη προσέγγιση δημιουργείται το φαινόμενο του false positives. Δηλαδή μπορεί να γίνονται λάθος συστάσεις σε ένα χρήστη αφού οι υπογραφές δύο χρηστών μπορεί να δείξουν ότι δύο τυχαίοι χρήστες είναι όμοιοι, όμως αυτό να προέκυψε από διαφορετικά tokens. Δηλαδή διαφορετικά token έχουν γυρίσει άσσους μέσα στην υπογραφή που όταν ελέγχονται επειδή βρίσκονται κοινοί τότε υπολογίζει το Jaccard Similarity ότι οι υπογραφές είναι σε μεγάλο βαθμό όμοιες.

Ο ακριβής αριθμός του μεγέθους των υπογραφών SIGNATURE_SIZE, του πόσες συναρτήσεις κατακερματισμού θα χρησιμοποιούνται ανάλογα με το μέγεθος των υπογραφών, το λιγότερο επιτρεπτό όριο ομοιότητας που πρέπει να έχουν δύο οποιοιδήποτε servers για να μπορούν να ανταλλάξουν χρήστες, καθώς επίσης και το λιγότερο επιτρεπτό όριο ομοιότητας που πρέπει να έχει ένας χρήστης σε σχέση με ένα server για να μπορεί να σταλεί ως μέλος του R_{s_i,s_j} προς το συγκεκριμένο server είναι κάποιοι παράγοντες που θα απασχολήσουν και τα πειράματα που θα γίνουν στη συνέχεια.

Η δεύτερη προσέγγιση παρουσιάζει πρόβλημα στο συνολικό χρόνο εκτέλεσης και αυτό οφείλεται στην ανταλλαγή χρηστών που δεν είναι απαραίτητοι (false positives). Η πολυπλοκότητα για τη πρώτη προσέγγιση όσων αφορά το κόστος επικοινωνίας φαίνεται να είναι πολύ ποιο μικρή σε σχέση με τη δεύτερη προσέγγιση. Αυτό οφείλεται στο μεγάλο

μέγεθος που έχουν οι χρήστες λόγω της υπογραφής που πρέπει να πηγαίνει μαζί τους όταν ανταλλάζονται στους εξυπηρετητές. Πρόβλημα επίσης υπάρχει και στην πολυπλοκότητα που εμφανίζουν οι αλγόριθμοι στη τελευταία φάση εκτέλεσης που είναι η σύσταση. Πάλι και σε αυτή τη περίπτωση φαίνεται ότι ο πρώτη προσέγγιση θα έχει καλύτερους χρόνους από τη πρώτη λόγω της πολυπλοκότητας.

Στην μεν πρώτη προσέγγιση για μια τυχαία κατανομή rayz οι χρήστες έχουν τυχαία ενδιαφέροντα. Άρα οι δομές που αφορούν τα token δεν θα είναι υπερφορτωμένες με πολλούς χρήστες. Αυτό μπορεί να αποδεικτέα χρησιμοποιώντας μαθηματικούς τύπους των πιθανοτήτων.

Πολυπλοκότητα Πρώτης Προσέγγισης

$$O\ (U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser\ *\ H_{p_{u_i}} * SearchToken + \sum_{i=1}^{m} Common_{Tokens} * Users_{per_{token}} * Cost_{per_{user}} + U_{s_i} * \boldsymbol{T_{u_i}} * Users_{per_{token}})$$

Πολυπλοκότητα Δεύτερης Προσέγγισης

$$O\ (U_{s_i} * processing_{cost\left(\boldsymbol{P_{u_i}}\right)} * SearchUser\ * H_{p_{u_i}} * Hashing_{token} + \sum_{j=1}^{m} R_{s_i,s_j} + U_{s_i} * (U_{s_i} + \sum_{j=1}^{m} R_{s_j,s_i}) \quad ,$$

$$R_{s_i,s_j} = users\ from\ i\ to\ j\ which\ have\ sim\_user\_server_{u_i,s_j} > threshold\_user,\ sim\_server_{s_i,s_j}\ ,\quad s_i \neq s_i$$

Σε ένα σύνολο με T λέξεις, U είναι όλοι οι χρήστες, και S όλα τα ενδιαφέροντα που λέει στα μηνύματα του ένας χρήστης ορίζεται η εξής πιθανότητα:

Η πιθανότητα για ένα τυχαίο χρήστη να αναφέρει μια συγκεκριμένη λέξη είναι $p = \frac{1}{T}$. Για όλους τους χρήστες του συνόλου U αν θέλουμε να βρούμε την ίδια πιθανότητα θα ισχύει $p = (\frac{1}{T})^U$. Άρα για T = 100,000 , U = 100,000 το p = 0 .Άρα καθώς αυξάνονται οι χρήστες μειώνεται η πιθανότητα ένα token να λαμβάνει μεγάλο ενδιαφέρον. Επίσης η πιθανότητα για

ένα χρήστη ως προς τα συνολικά του ενδιαφέροντα S να έχει αναφέρει μια συγκεκριμένη λέξη είναι $p = \frac{\binom{T}{S-1} * \binom{T}{1}}{\binom{T}{S}}$.

Αυτός είναι ένας λόγος που πειραματικά η μελέτη της πρώτης προσέγγισης έδειξε καλύτερα αποτελέσματα. Στη δεύτερη προσέγγιση δύσκολα μπορούμε να γλυτώσουμε το $O(n^2)$. Διότι είναι μια n x n αναζήτηση στην οποία πρέπει για όλους τους χρήστες να γίνει έλεγχος με όλους τους υπόλοιπους χρήστες. Ενώ στη πρώτη προσέγγιση για το κάθε χρήστη θα ελέγξουμε μέσα στα token του και θα βρούμε μόνο τους χρήστες που είπαν και αυτοί αυτό το token. Άρα όπως έχει αποδειχθεί ποιο πάνω το $T_{u_i} * Users_{per_{token}}$ δεν θα φτάσει ποτέ το $U_{s_i} + \sum_{j=1}^{m} R_{s_j,s_i}$ που ισχύει στη δεύτερη προσέγγιση.

Με την ποιό πάνω επεξήγηση, γίνεται κατανοητό το ότι το αποτέλεσμα για τις συστάσεις προκύπτει σαφώς ποιο γρήγορα στη πρώτη προσέγγιση. Στη συνέχεια θα πρέπει να εξετάσουμε τι γίνεται με το κόστος επικοινωνίας των εξυπηρετητών στις δύο περιπτώσεις. Για να λεχθούν επεξηγηματικά σχόλια θα πρέπει να γίνουν πειράματα για να αποδειχθεί πειραματικά τι συμβαίνει. Εκ πρώτης όψεως η δεύτερη προσέγγιση επειδή στο χρήστη πρέπει να μεταφέρεται η κατάσταση του αυτό θα δείξει αυξημένο κόστος επικοινωνίας σε σχέση με τη πρώτη προσέγγιση που το μόνο που μεταφέρεται εκτός του ονόματος του είναι ένα ακέραιος που δηλώνει το βάρος σε ένα συγκεκριμένο token.

Η απόδειξη των όσων έχουν λεχθεί και επεξηγηθεί θα γίνει σε επόμενο κεφάλαιο που θα παρουσιαστούν τα πειράματα με πραγματικές τιμές.

Κεφάλαιο 5

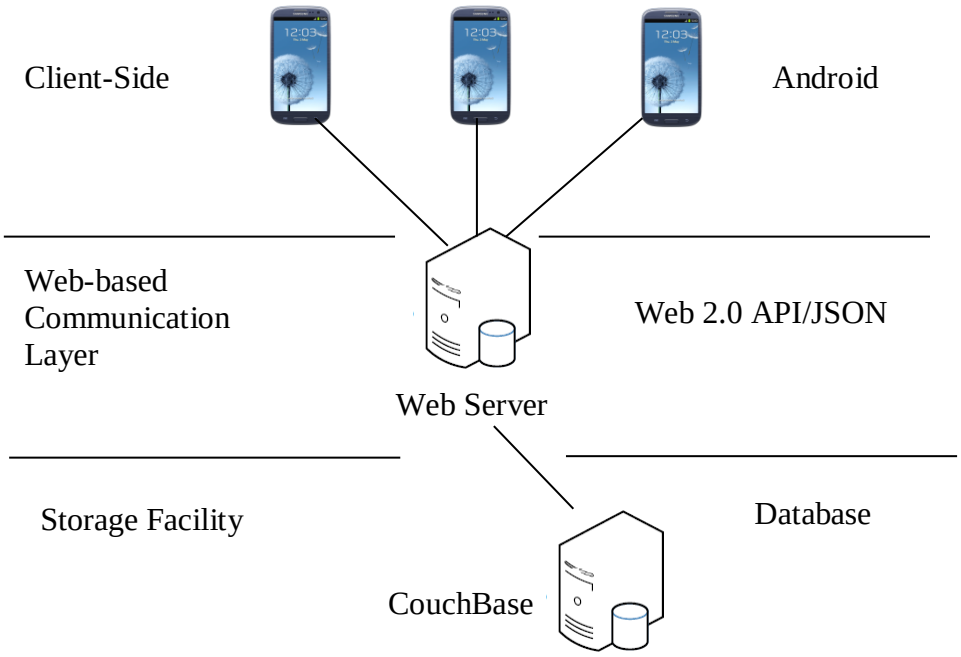
Πρωτότυπο Σύστημα Υποστήριξης Πλατφόρμας Rayzit

Σε αυτό το κεφάλαιο θα γίνει περιγραφή ενός πρωτότυπου συστήματος που έχει υλοποιηθεί στην πλατφόρμα Android, για να υποστηρίξει την υπηρεσία του «Rayzit». Στη πρώτη υποενότητα θα αναφερθούν κάποια γενικά στοιχεία της εφαρμογής και στην επόμενη υποενότητα θα γίνει παρουσίαση κάποιων οθονών που παρουσιάζονται στο χρήστη και πως μέσα από αυτές μπορεί να γίνει η πλοήγηση στην εφαρμογή.

5.1 Εισαγωγή

Η εφαρμογή όπως έχει αναφερθεί έχει υλοποιηθεί στη πλατφόρμα Android χρησιμοποιώντας το λειτουργικό Android OS και εκτελέσιμο το αρχείο (δηλαδή το .APK) είναι 3,29MB. Ο κώδικας είναι γραμμένος σε γλώσσα JAVA και απαριθμεί 15,000 γραμμές κώδικα (Lines of Code - LOC). Επίσης 2,900 LOC έχουν χρησιμοποιηθεί για XML αρχεία που καθορίζουν τη γραφική παρουσίαση της εφαρμογής. Η εφαρμογή αναπτύχθηκε για να ολοκληρώσει το σύστημα συστάσεων που έχει προταθεί μέσα από τον αλγόριθμο SNA και οι χρήστες που το χρησιμοποιούν να λαμβάνουν τις συστάσεις που προκύπτουν από τον αλγόριθμο.

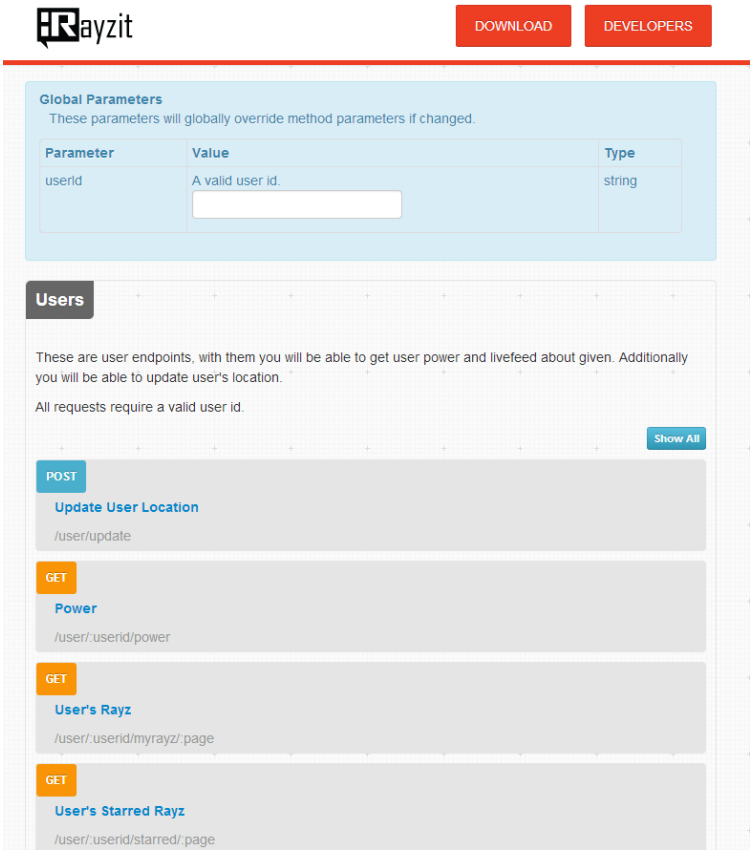
Όπως φαίνεται στο Σχήμα 5.1 η αρχιτεκτονική υποβολής και παραλαβής μηνυμάτων για τους χρήστες της εφαρμογής χωρίζεται σε 3 στάδια. Στο πρώτο στάδιο παρουσιάζεται η επικοινωνία του απλού χρήστη χρησιμοποιώντας το $2^{ο}$ επίπεδο που είναι ένα Web-based πλαίσιο επικοινωνίας στο οποίο χρησιμοποιείται τεχνολογία διαδικτύου 2.0 (Web 2.0 API).



Σχήμα 5.1. Η αρχιτεκτονική υποβολής και παραλαβής μηνυμάτων για τους χρήστες που χρησιμοποιούν το «Rayzit»

Ο όρος Web 2.0 χρησιμοποιείται για να περιγράψει τη νέα γενιά του Παγκόσμιου Ιστού η οποία βασίζεται στην όλο και μεγαλύτερη δυνατότητα των χρηστών του διαδικτύου να μοιράζονται πληροφορίες και να συνεργάζονται online. Αυτή η νέα γενιά είναι μια δυναμική πλατφόρμα στην οποία μπορούν να αλληλεπιδρούν χρήστες χωρίς ειδικευμένες γνώσεις σε θέματα υπολογιστών και δικτύων. Συγκεκριμένα βασίζεται σε ένα νέο τρόπο σχεδίασης ιστοσελίδων ο οποίος θα βασίζεται στη διάδραση του χρήστη. Θα επιτρέπει στον χρήστη να αλλάξει τόσο το περιβάλλον της σελίδας όσο και να παρέμβει στο περιεχόμενό της.

Παράδειγμα του Web 2.0 API/JSON φαίνεται στο Σχήμα 5.2. Ο προγραμματιστής μπορεί να χρησιμοποιήσει μια από τις επιλογές που φαίνονται και στο Σχήμα 5.2 και να εξάγει πληροφορίες για ένα χρήστη του συστήματος οι οποίες θα έρθουν σε μορφή JSON. Στο Σχήμα 5.3 φαίνεται ένα παράδειγμα για το πώς η ημι-δομημένη πληροφορία JSON λαμβάνεται από την εφαρμογή όπου θα παρουσιαστεί στο χρήστη όπως θα φανεί σχηματικά στην επόμενη υποενότητα μέσα από τις οθόνες.

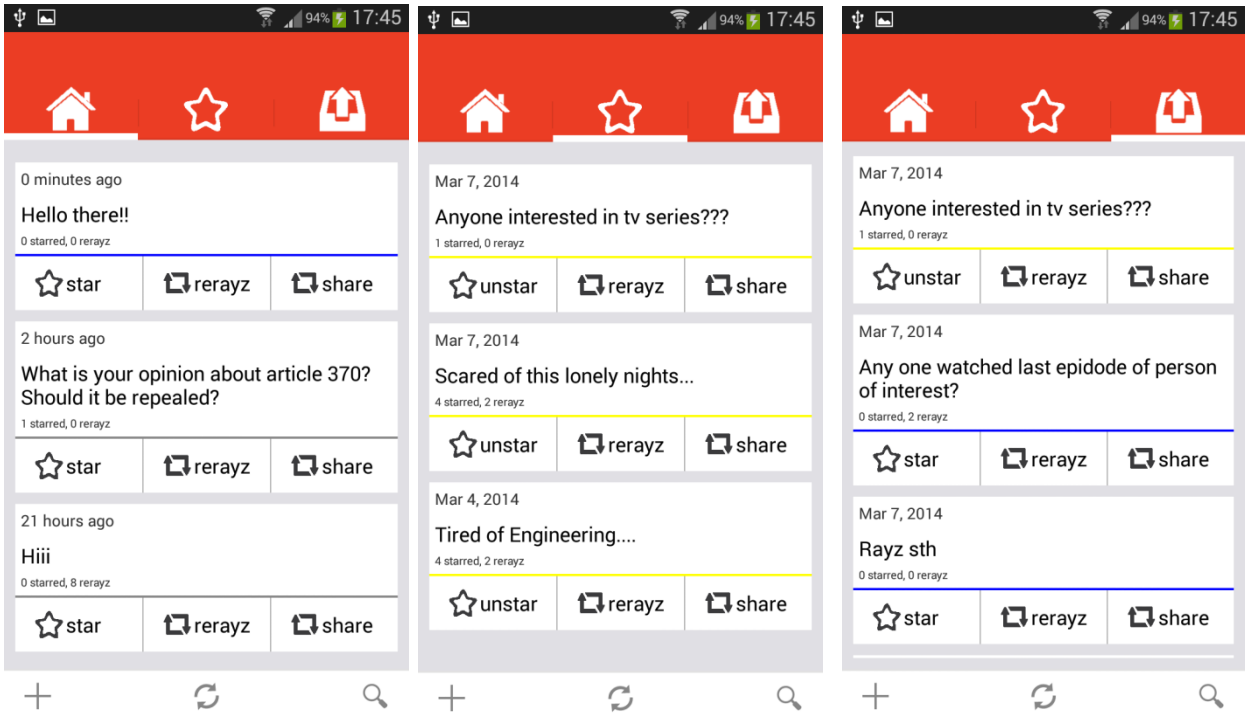


Σχήμα 5.2. Web 2.0 API/JSON του Rayzit

5.1 Γραφικό Περιβάλλον Διπροσωπίας

Ένα καλό περιβάλλον διπροσωπίας είναι αυτό που βοηθά ένα χρήστη στη πλοήγηση του καθώς χρησιμοποιεί την εφαρμογή περιεργάζεται τις διάφορες λειτουργίες της. Γι' αυτό όπως φαίνεται στις οθόνες που παρουσιάζονται έχει επιλεγεί η εφαρμογή να έχει το δικό της «κατοχυρωμένο» χρώμα όπως γίνεται και με άλλες δημοφιλείς εφαρμογές που είναι π.χ το Viber που έχει χρώμα μοβ, είτε το Facebook που έχει χρώμα μπλε. Στο Rayzit έχει επιλεγεί το χρώμα κόκκινο για να γίνεται η αντίθεση.

Στην πρώτη οθόνη (αριστερά προς δεξιά) που φαίνεται στο Σχήμα 5.4 είναι η πρώτη οθόνη που βλέπει αρχικά ο χρήστης μόλις ανοίξει την εφαρμογή. Αυτή οθόνη ονομάζεται Live Feed (Home Picture) και σε αυτή βρίσκονται όλα τα rayz τα οποία κατά καιρούς ο χρήστης έχει παραλάβει. Τα rayz που υπάρχουν σε αυτό το τμήμα δεν ανέρχονται πάνω από ένα συγκεκριμένο αριθμό (100) και δεν θα πρέπει να έχουν ημερομηνία δημιουργίας μεγαλύτερης του ενός μήνα. Όταν κάποιο rayz εμπίπτει στις δύο κατηγορίες τότε διαγράφεται από το Live Feed. Αυτός είναι άλλωστε και ο σκοπός του να έχει τα ποιο πρόσφατα σε αυτό το τμήμα της εφαρμογής.



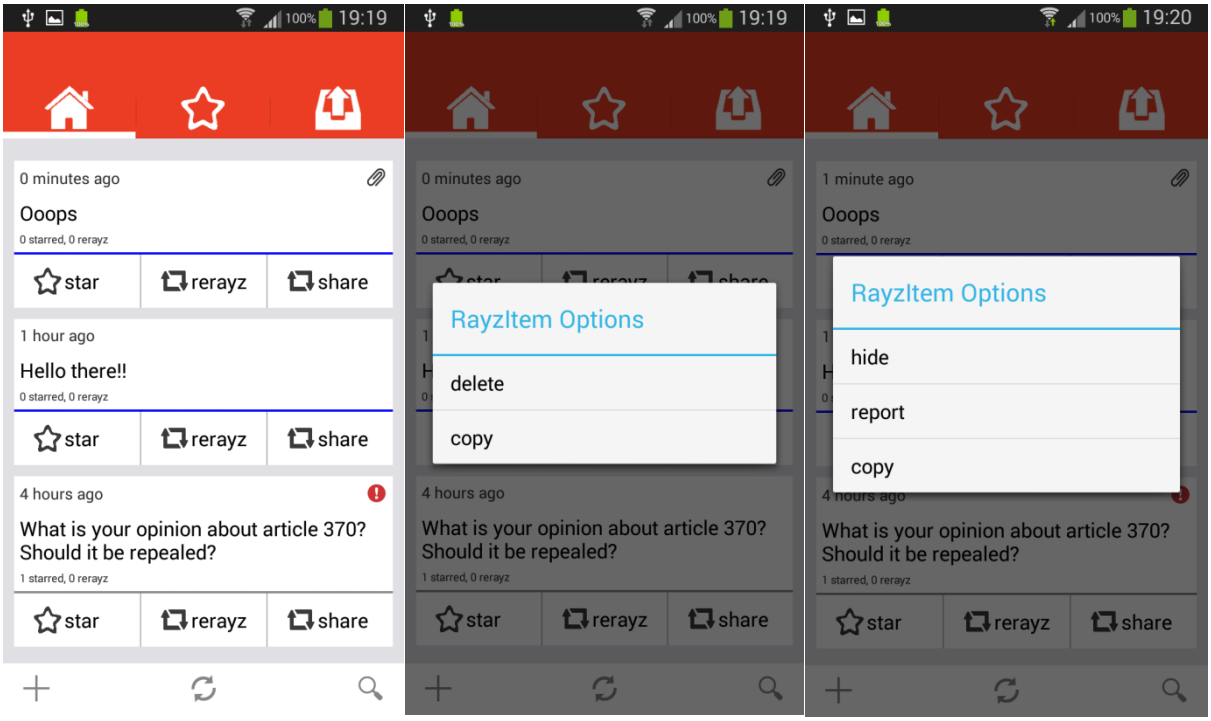
Σχήμα 5.3. Οθόνες της εφαρμογής Rayzit. Η πρώτη (αριστερά προς δεξιά) απεικονίζεται το Live Feed στη δεύτερη το Starred Rayz και στη τρίτη οθόνη το My Rayz

Το δεύτερο τμήμα το οποίο απεικονίζεται με ένα αστέρι, είναι το τμήμα στο οποίο ο χρήστης έχει αποθηκευμένα τα rayz τα οποία είναι τα αγαπημένα του, και στα οποία θέλει να λαμβάνει ενημερώσεις. Οι ενημερώσεις αφορούν στο πόσοι άλλοι έχουν επιλέξει να είναι και για αυτούς αγαπημένο rayz ή πόσοι έχουν επιλέξει να το στείλουν στουw kSNN γείτονες τους (βλέπε 3η ετικέτα σε κάθε rayz π.χ «1 starred , 0 rerayz»).

Η λειτουργία του star και του rerayz γίνεται όταν ο χρήστης πατήσει την αντίστοιχη επιλογή όπως αυτή φαίνεται στις οθόνες. Όταν σε κάποιο rayz πατηθεί το star τότε μεταφέρεται και στη λίστα των Starred Rayz στο 2ο τμήμα της εφαρμογής. Γι' αυτό και στο τρίτο τμήμα της εφαρμογής που είναι το My Rayz όπου υπάρχουν τα μηνύματα που έχει στείλει ο χρήστης βλέπουμε ότι το πρώτο rayz που απεικονίζεται βρίσκεται και στο 2ο τμήμα. Συνεπώς το rayz αυτό έχει δημιουργηθεί από το χρήστη που ο ίδιος επέλεξε να είναι στα αγαπημένα του και να λαμβάνει ενημερώσεις. Επίσης σε όλα τα rayz απεικονίζεται και η ημερομηνία δημιουργίας των rayz.

Στο Σχήμα 5.5 φαίνονται τρία διαφορετικά rayz. Στο πρώτο το σύμβολο του συνημμένου εγγράφου δηλώνει ότι το rayz αυτό έχει συνημμένο εικόνα ή/και βίντεο, ή/και ηχογράφηση. Στο τρίτο rayz το σύμβολο του κινδύνου δείχνει ότι κάποιος χρήστης θεώρησε ακατάλληλο

το περιεχόμενο του rayz και το έχει αναφέρει στην υπηρεσία για να το δει κάποιος υπεύθυνος και να το αφαιρέσει.

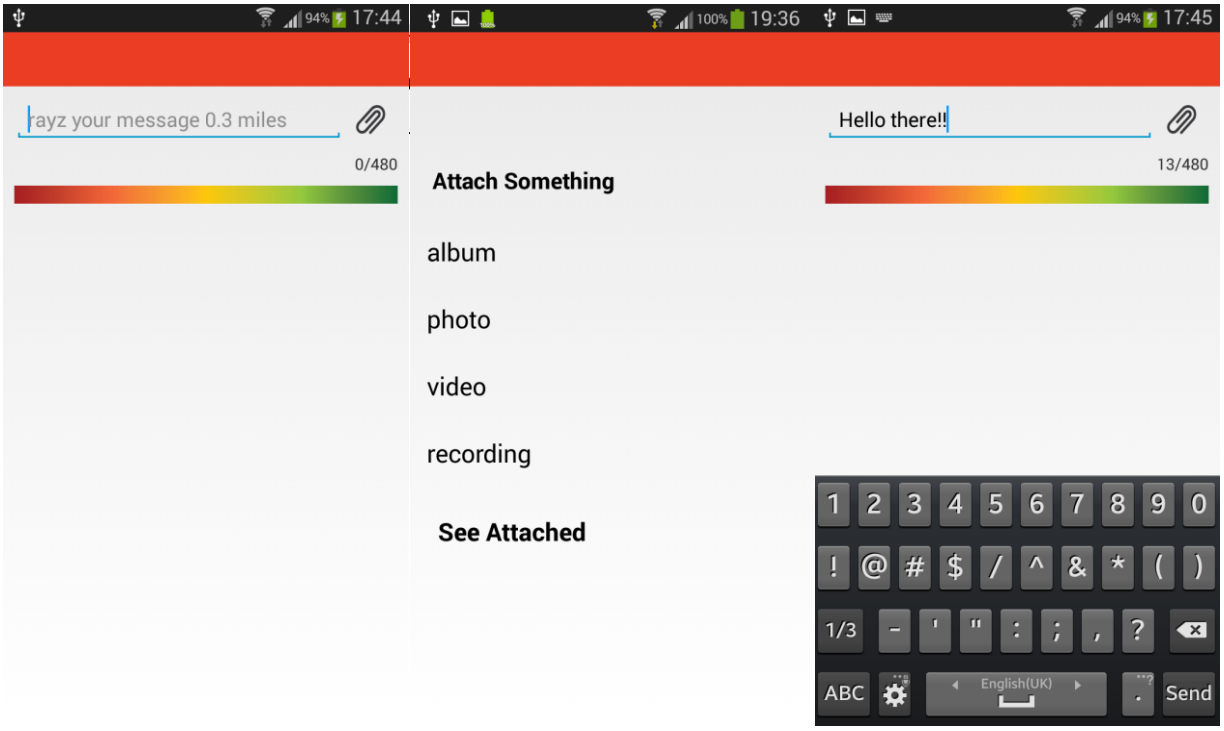


Σχήμα 5.4. Οθόνες της εφαρμογής Rayzit. Η πρώτη (αριστερά προς δεξιά) απεικονίζει 2 είδους διαφορετικά rayz. Το πρώτο είναι My Rayz και το τρίτο Stranger Rayz. Η δεύτερη και Τρίτη εικόνα δείχνουν τις αντίστοιχες επιλογές για αυτά τα Rayz

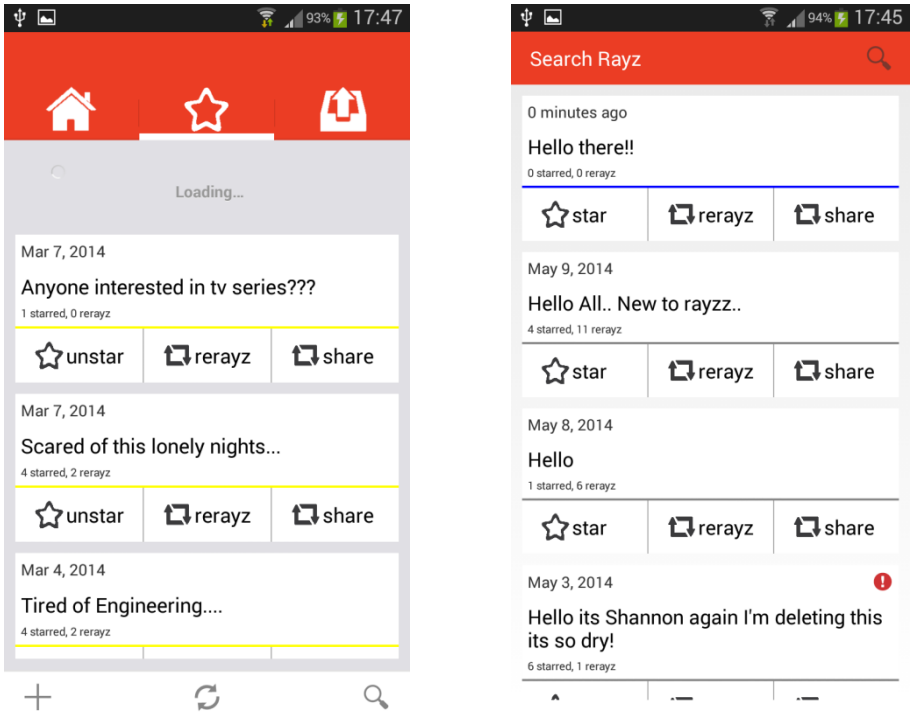
Στη δεύτερη εικόνα στο Σχήμα 5.5 είναι οι επιλογές οι οποίες ισχύουν για το πρώτο rayz που είναι το rayz έχει δημιουργηθεί από το χρήστη. Υπάρχουν οι επιλογές της διαγραφής και της αντιγραφής του περιεχομένου του rayz. Αντίστοιχα στη Τρίτη εικόνα είναι οι επιλογές που εμφανίζονται για το τρίτο rayz που είναι ενός αγνώστου. Οι επιλογές είναι να γίνει απόκρυψη που είναι ουσιαστικά η διαγραφή και η μη λήψη ενημερώσεων για αυτό και η αναφορά ακατάλληλου περιεχομένου καθώς και η αντιγραφή περιεχομένου.

Ο χρήστης όταν πατήσει το πρώτο σύμβολο (+) όπως αυτό φαίνεται στο Σχήμα 5.5 τότε μεταφέρεται στην πρώτη οθόνη που φαίνεται στο Σχήμα 5.6 όπου εκεί μπορεί να πληκτρολογήσει το μήνυμα του. Αν επιλέξει την επιλογή του συνημμένου που φαίνεται στην πρώτη οθόνη στο Σχήμα 5.6 πάνω αριστερά μεταφέρεται στη δεύτερη οθόνη όπου εκεί μπορεί να επιλέξει μεταξύ φωτογραφίας από τη γκαλερί του κινητού του, να βγάλει φωτογραφία ή βίντεο εκείνη των ώρα ή να ηχογραφήσει συνομιλία. Δεν περιορίζεται στη χρήση μόνο μιας από αυτές τις επιλογές. Στο τέλος πατώντας στο send που φαίνεται στη τρίτη οθόνη αποστέλλεται το μήνυμα του στου kSNN γείτονες του.

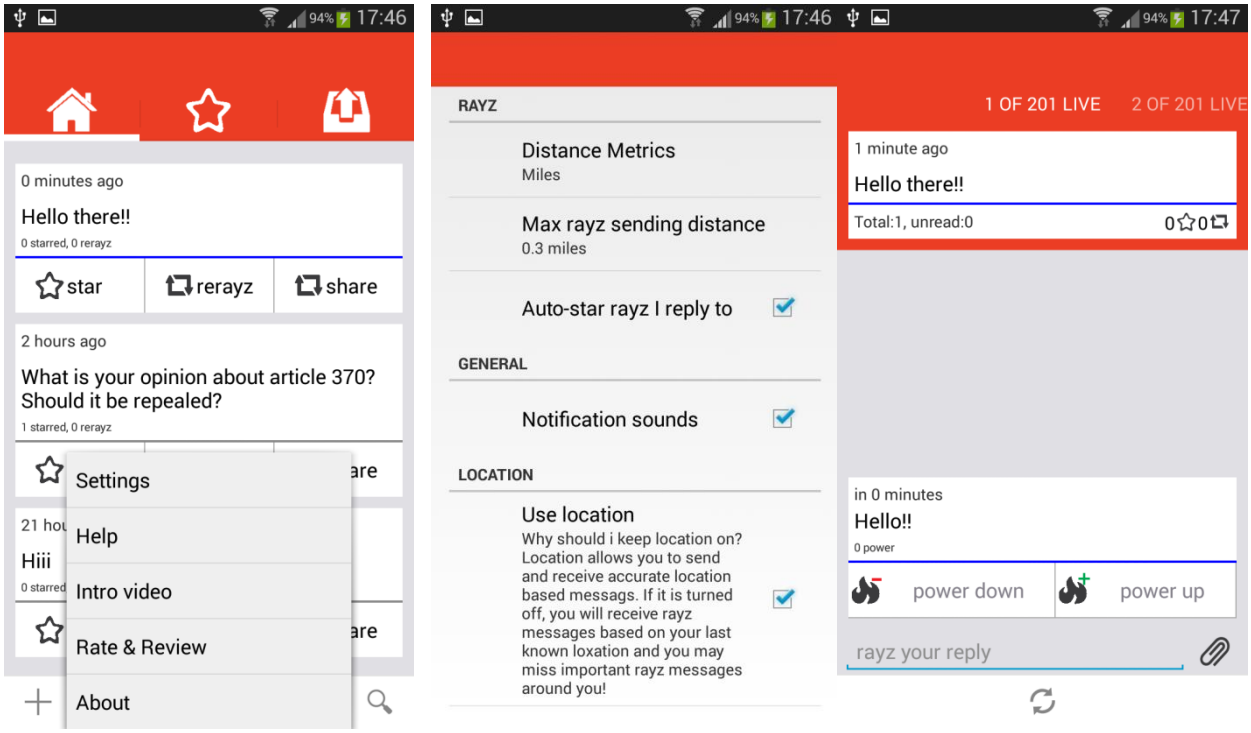
Στη πρώτη οθόνη φαίνεται η χρήση της ενέργειας (χρωματιστή ένδειξη μπάρας κάτω από την είσοδο κειμένου). Ο χρήστης για να στείλει κάποιο rayz του αποκόπτεται ενέργεια. Έτσι εαν δεν έχει ενέργεια δεν μπορεί να χρησιμοποιήσει το Rayzit. Το πώς ανακτάται και χάνεται η ενέργεια θα αναφερθεί ποιο κάτω.



Σχήμα 5.5. Οθόνες της εφαρμογής Rayzit. Η πρώτη (αριστερά προς δεξιά) το περιβάλλον που χρησιμοποιεί ο χρήστης για να στείλει ένα rayz . Στη δεύτερη είναι οι επιλογές που παρέχονται στο χρήστη για να στείλει πολυμέσα στο μήνυμα του και στη τρίτη εικόνα ο τρόπος αποστολής



Σχήμα 5.6. Οθόνες της εφαρμογής Rayzit. Η πρώτη (αριστερά προς δεξιά) είναι η λειτουργία του Refresh όπου κατεβάζει τα rayz των kSNN χρηστών για το χρήστη και στη δεύτερη οθόνη είναι η λειτουργία της αναζήτησης των rayz με μια λέξη κλειδί



Σχήμα 5.7. Οθόνες της εφαρμογής Rayzit. Η πρώτη (αριστερά προς δεξιά) είναι η λειτουργία οι επιλογές που παρέχονται και στη δεύτερη οθόνη οι ρυθμίσεις. Στη τρίτη εικόνα φαίνονται οι απαντήσεις για ένα rayz.

Στη πρώτη οθόνη στο Σχήμα 5.8 είναι οι επιλογές που παρέχει η υπηρεσία στο πελάτη της εφαρμογής και όταν γίνει η επιλογή Settings εμφανίζεται η οθόνη στα δεξιά. Ο χρήστης έχει τη δυνατότητα να επιλέξει το kSNN να γίνεται με βάση μίλια ή χιλιόμετρα, ακόμα να επιλέξει το εύρος μέσα στο οποίο ο αλγόριθμος θα ψάξει να βρει το σύνολο kSNN. Ακόμα παρέχεται η επιλογή το κάθε μήνυμα που στέλνεται από το χρήστη στην υπηρεσία αυτόματα να γίνεται και starred. Επίσης ο χρήστης μπορεί να απενεργοποιήσει και να ενεργοποιήσει τους ήχους που ακούγονται όταν στέλνει ή απαντά ένα μήνυμα.

Στη τρίτη εικόνα του σχήματος 5.8 είναι ένα παράδειγμα απαντήσεων για κάποιο rayz. Ο χρήστης μπορεί χρησιμοποιώντας το rayz your reply να πληκτρολογήσει την απάντηση του και ακολούθως το μήνυμα θα σταλεί σε αυτόν που έχει στείλει εξ' αρχής το rayz. Στο πάνω μέρος φαίνεται η ένδειξη «1 of 201 Live» όπου αυτό δίνει στο χρήστη μια ένδειξη για το που βρισκόταν προηγουμένως όταν πάτησε πάνω στο rayz για να δει τις απαντήσεις. Ο χρήστης μπορεί να εισάγει πολυμέσα όπως και σε ένα απλό μήνυμα και με την επιλογή του refresh (όπως φαίνεται στο κέντρο στο κάτω μέρος της εικόνας) να ζητήσει τις ανανεωμένες απαντήσεις για αυτό το rayz.

Για όλα τις απαντήσεις που δίνονται υπάρχει η επιλογή power up και power down. Ο χρήστης με το power up επιβραβεύει τη συγκεκριμένη απάντηση ως σχετική με το θέμα συζήτησης που είναι το μήνυμα rayz (Hello There!!). Οπόταν στη περίπτωση της

επιβράβευσης θα δοθεί ενέργεια ως έπαθλο στο χρήστη που έστειλε αυτή την απάντηση ενώ σε αντίθετη περίπτωση θα αφαιρεθεί. Άρα με αυτό τον τρόπο προωθείται στους χρήστες η ιδέα να μην χρησιμοποιούν την εφαρμογή επιπόλαια γιατί με αυτό τον τρόπο θα χάνουν ενέργεια και δεν θα μπορούν να στείλουν rayz.

Κεφάλαιο 6

Πειραματική Μεθοδολογία

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τη πειραματική μεθοδολογία που χρησιμοποιήσαμε για να εξετάσουμε τη συμπεριφορά του SN Algorithm σε σχέση με συγκεκριμένες παραμέτρους, αρχιτεκτονική και υπολογιστική δύναμη που υπήρχε διαθέσιμη κατά τη διεξαγωγή των πειραμάτων.

Στην αρχή θα παρουσιαστούν και θα επεξηγηθούν με λεπτομέρεια οι λόγοι που ώθησαν να εξεταστεί ο συγκεκριμένος αλγόριθμος υπό αυτές τις συνθήκες, δηλαδή θα γίνει πλήρως αντιληπτό το τι περιμένουμε να δούμε σε αυτά τα πειράματα εξετάζοντας τις συγκεκριμένες παραμέτρους αλλά και θα γίνει ξεκάθαρο γιατί έγινε η επιλογή για τα συγκεκριμένα πειράματα.

6.2 Αρχιτεκτονική Υποδομή Διεξαγωγής Πειραμάτων

Σε αυτή την υποενότητα θα επεξηγηθεί η Αρχιτεκτονική Υποδομή Διεξαγωγής Πειραμάτων. Θα αναλυθεί πλήρως το πραγματικό υλικό (Hardware) που χρησιμοποιήθηκε για την διεξαγωγή πειραμάτων και συγκεκριμένα θα δοθούν πλήρης στοιχεία που αφορούν τα χαρακτηριστικά που διέπουν την αρχιτεκτονική του DMSL VCenter datacenter[1].

Συγκεκριμένα ο DMSL απαρτίζεται από 5 IBM System x3550 M3 και HP Proliant DL 360 G7 συστάδες υπολογιστών οι οποίοι απαρτίζονται από single socket (8 cores) ή dual socket (16 cores) Intel(R) Xeon(R) CPU E5620 @ 2.40GHz, αντίστοιχα. Ολόκληρη η υποδομή διασυνδέεται με δίκτυο μετάδοσης δεδομένων της τάξης των Gigabit, ενώ συνολικά η υποδομή συναποτελείται από 300GB κύριας μνήμης και από ένα αποθηκευτικό χώρο που απαριθμεί 16TB από RAID-5 και SSD μονάδων αποθήκευσης πάνω στον IBM 3512. Ολόκληρη η υποδομή διευθύνεται από ένα VMWare vCenter Server 5.1 ο οποίος διευθύνει αντίστοιχα τους VMWare ESXi (OS).

6.3 Υπολογιστική Υποδομή Μονάδων Επεξεργασίας (Server's Features)

Σε αυτή την υποενότητα θα επεξηγηθεί η Υπολογιστική Υποδομή των Μονάδων Επεξεργασίας, δηλαδή θα εξηγηθεί από το λειτουργικό σύστημα που χρησιμοποιήθηκε για τη διεξαγωγή των πειραμάτων και θα δοθούν αναλυτικά τα συστατικά που το αποτελούν.

Οι εικονικές υπολογιστικές μονάδες (virtual machines) που χρησιμοποιήθηκαν για την διεξαγωγή των πειραμάτων με το υλικό (hardware) που επεξηγήθηκε στη προηγούμενη ενότητα, μαζί συναποτελούν το εικονικό cluster που κατέχει το DMSL εργαστήριο του Πανεπιστημίου Κύπρου και πάνω στο οποίο έγιναν τα πειράματα.

Συγκεκριμένα ολόκληρη η εικονική συστάδα ονομάζεται VCenter IaaS datacenter. Αποτελείται από 9 Ubuntu Server 12.04 images (η κάθε υπολογιστική μονάδα – server – έχει ήδη δηλωθεί σε προηγούμενο σημείο ως s_i) όπου ο καθένας έχει 8GB κύριας μνήμης (RAM) με 2 εικονικές κεντρικές μονάδες επεξεργασίας (KME - CPU) (@ 2.40 GHz).

Οι υπολογιστικές μονάδες χρησιμοποιούν για τα ενδιάμεσα αποτελέσματα που προκύπτουν 10K RPM LSILogic SCSI δίσκους οι οποίοι έχουν διαμορφωθεί με VMFS 5.54 (1MB block

size). Πάνω σε κάθε υπολογιστική μονάδα έχει εγκατασταθεί η Parallel Java[2] βιβλιοθήκη για να μπορούμε να χρησιμοποιήσουμε τις διάφορες βασικές της λειτουργίες για τη σωστή διεξαγωγή των πειραμάτων. Η ποιο βασική λειτουργία που προσφέρει η Parallel Java είναι το MPI message passing που χρησιμοποιείται εκτενώς στον SNA όπως επεξηγήθηκε σε προηγούμενο κεφάλαιο για την ανταλλαγή μηνυμάτων μεταξύ των εξυπηρετητών (server).

Το MPI είναι ένα πρωτόκολλο επικοινωνίας για κατανεμημένα παράλληλα προγράμματα που τρέχουν πάνω σε διάφορες μονάδες επεξεργασίας, είτε αυτοί είναι απλοί προσωπικοί υπολογιστές είτε είναι εξυπηρετητές, και προσφέρει επικοινωνία τύπου «σημείο-με-σημείο» (point-to-point) μεταξύ των μονάδων επεξεργασίας. Χρησιμοποιείται ανεξαρτήτου γλώσσας προγραμματισμού και έχει σαν στόχο την υψηλή απόδοση, την επεκτασιμότητα (scalability) και την φορητότητα (portability) [paper here].

6.4 Υπηρεσία «Rayzit»

Σε αυτή την υποενότητα θα επεξηγηθεί η υπηρεσία «Rayzit» και πως μας βοηθά στο να διεξάγουμε τα πειράματα του SR Algorithm. Θα επεξηγηθεί συνοπτικά που βασίζει την λειτουργία της αυτή η υπηρεσία και πως γίνεται η αποθήκευση των δεδομένων τα οποία χρησιμοποιούνται στα πειράματα.

Η υπηρεσία Rayzit είναι μια υπηρεσία ανταλλαγής μηνυμάτων, σχεδιασμένη να δουλεύει αποδοτικά ακόμα και όταν χρησιμοποιείται από μεγάλα πλήθη (crowd messaging service), η οποία και επεξηγήθηκε εκτενώς σε προηγούμενο κεφάλαιο. Η εφαρμογή είναι ρυθμισμένη σε μία «αποστρατικοποιημένη» ζώνη δίκτυο (demilitarized network zone) η οποία αποτελείται από ένα HAProxy HTTP load balancer για να μπορεί να κατανέμει ισάξια στους αντίστοιχους εξυπηρετητές φόρτο εργασίας. Ισάξια γίνεται επίσης αντίστοιχα ο διαμοιρασμός των φυσικών πόρων του datacenter όπως αυτοί αναφέρθηκαν στη ποιο πάνω υποενότητα.

Ο κάθε server χαρακτηρίζεται από ένα Apache HTTP server, μια Couchbase NoSQL document store[4] για την αποθήκευση των μηνυμάτων που στέλνουν οι χρήστες της εφαρμογής μεταξύ τους. Στη Couchbase, τα δεδομένα αποθηκεύονται σε μορφή JSON όπως έχει επεξηγηθεί και σε προηγούμενο κεφάλαιο. Η συγκεκριμένη μορφή μας επιτρέπει να κάνουμε γρήγορη ευρετηρίαση των δεδομένων που έχουμε συλλέξει και έτσι να παίρνουμε γρήγορα τα αποτελέσματα που μιας επερώτησης. Επίσης εύκολα μπορούμε ακόμα και άμεσα να πάρουμε δεδομένα χρησιμοποιώντας το Rayzit Web 2.0 API.

Η βάση αυτή χρησιμοποιείται πειραματικά για την ανάκτηση των μηνυμάτων των χρηστών της υπηρεσίας Rayzit έτσι ώστε να μπορούμε κάθε μερικά δευτερόλεπτα να εξάγουμε νέα αποτελέσματα.

Ο αλγόριθμος SNA, συνοπτικά, κάθε ένα χρονικό διάστημα t, εκτελείται πάνω στους εξυπηρετητές, όπως επεξηγήθηκε σε προηγούμενη ενότητα και για κάθε πιθανό χρήστη που υπάρχει καταχωρημένος μέσα στο σύστημα θα προτείνει σε αυτόν πιθανούς χρήστες οι οποίοι βρίσκονται «κοντά» στις προτιμήσεις του. Να γίνει αντιληπτό ότι με τον όρο «κοντά», όπως επεξηγήθηκε και σε προηγούμενο κεφάλαιο εννοούμε πάντα ότι οι χρήστες βρίσκονται κοντά όσων αφορά την γεωτοποθέτηση τους στο πραγματικό κόσμο αλλά και κοντά όσων αφορά τις προτιμήσεις που κατά καιρούς εξέφρασαν μέσα από τα μηνύματα που έχουν ανταλλάξει αναμεταξύ τους.

6.5 Δεδομένα που χρησιμοποιήθηκαν

Θέλοντας να αξιολογηθεί πλήρως η συμπεριφορά του αλγορίθμου θα πρέπει να χρησιμοποιηθούν δεδομένα που είναι κοντά στην πραγματικότητα δηλαδή για να δείξουμε τη συμπεριφορά του αλγορίθμου σε τυχαία δεδομένα.

Θα πρέπει να εξετάσουμε ποιος είναι ο ιδανικός αριθμός από μηνύματα (g) που αντιστοιχούν για μια δεδομένη χρονική στιγμή t σε ένα χρήστη και επίσης ποιό είναι το ιδανικό μέγεθος μηνύματος το οποίο εμείς το μετράμε σε λέξεις αφού τα ενδιαφέροντα που έχει κάποιος χρήστης εξάγονται μέσα από αυτά που έχουν ειπωθεί από το χρήστη μέσα από τις λέξεις που έχει γράψει σε ένα του μήνυμα. Γι’ αυτό το λόγο επιλέγηκε να χρησιμοποιηθούν ρεαλιστικού τύπου δεδομένα.

Για να εξάγουμε συμπεράσματα που θα μας βοηθήσουν να κατανοήσουμε αυτές τις δύο μετρικές που αναφέρθηκαν στη προηγούμενη υποενότητα, θα πρέπει να μελετήσουμε ένα από μεγαλύτερους οργανισμούς που ασχολείται με κοινωνικά δίκτυα αλλά και που η φύση του κοινωνικού του δικτύου βρίσκεται πολύ κοντά στην υπηρεσία «Rayzit».

Ο οργανισμός που έχει επιλεγεί και βρίσκεται κοντά στην υπηρεσία «Rayzit» και μπορούμε να χρησιμοποιήσουμε κάποια στατιστικά στοιχεία της εν λόγω υπηρεσίας που έχουν είναι το Twitter. Σύμφωνα με στατιστικά στοιχεία, το Twitter κατά μέσο όρο κάθε μέρα δέχεται γύρω στα 58 εκατομμύρια tweets για περίπου 645,750,000 ενεργούς χρήστες. Αυτά τα νούμερα

πρέπει να προβληματίσουν για το ποία θα είναι η τιμή που θα πάρει η μεταβλητή g, για το ποιος θα είναι ο ιδανικός αριθμός που ένας μέσος χρήστης θα στείλει σε ένα χρονικό διάστημα t. Επίσης πρέπει να προβληματίσει και η υπολογιστική δύναμη που έχει το twitter σε σχέση με την υπολογιστική δύναμη που έχει αναφερθεί σε προηγούμενη υποενότητα.

Λαμβάνοντας υπόψη τους παράγοντες που μόλις αναφέρθηκαν η πλέον συμφέρουσα προσέγγιση, αν θεωρήσουμε ότι όλοι οι χρήστες, χρησιμοποιούν συνεχώς την υπηρεσία «Rayzit», κατά μέσο όρο ανά χρονική στιγμή t θα παράγουν περίπου 5-8 μηνύματα (rayz) άρα για 1,000,000 χρήστες θα έχουμε περίπου γύρω στα 4,000,000 μηνύματα. Αυτό το νούμερο είναι το πλέον αντιπροσωπευτικό και βρίσκεται πολύ κοντά στην πραγματικότητα αφού αναλογικά είναι πολύ κοντά στο νούμερο που αντιστοιχεί στα tweets που λαμβάνει κάθε μέρα το Twitter, αλλά είναι και ανάλογο της υπολογιστικής δύναμης που έχει το DMSLab.

Ο άλλος παράγοντας που θα πρέπει να ρυθμιστεί στα δεδομένα που θα χρησιμοποιηθούν είναι το μέγεθος των μηνυμάτων που κατά μέσο όρο στέλνει ένας μέσος χρήστης. Σύμφωνα με έρευνες έχει διαπιστωθεί ότι σε μία πρόταση περίπου αναλογούν γύρω στους 50 χαρακτήρες. Κατά μέσο όρο οι 50 αυτοί χαρακτήρες παράγουν γύρω στις 7.5 λέξεις κάθε φορά. Αξίζει να σημειωθεί ότι για το Twitter ισχύει ότι ένα μέσο tweet έχει περίπου 60 χαρακτήρες. Άρα θέλοντας και πάλι τα δεδομένα να βρίσκονται όσο ποιο κοντά γίνεται στη πραγματικότητα θεωρήσαμε ότι οι 60 χαρακτήρες παράγουν γύρω στις 8 λέξεις.

Εμπειρικά, μελετώντας την υπηρεσία «Rayzit», κάτι τέτοιο δεν ισχύει σε μεγάλο βαθμό. Οι πραγματικοί χρήστες του ενεργού συστήματος που χρησιμοποιείται τώρα χρησιμοποιούν από 5-10 λέξεις μέσα στα μηνύματα που ανταλλάζουν μεταξύ τους. Γι' αυτό και ορίζεται ως το ιδανικό εύρος για τη μεταβλητή len οι τιμές από 5 μέχρι 10 λέξεις το οποίο βρίσκεται κοντά στους 60 χαρακτήρες των tweet των χρηστών του Twitter.

Συνοψίζοντας θα πρέπει να αναθεωρήσουμε ότι για τα πραγματικά τυχαία δεδομένα θα χρησιμοποιηθούν 3-8 μηνύματα για κάθε χρήστη και το κάθε μήνυμα θα έχει εύρος από 5-8 λέξεις.

6.6 Παράμετροι

Σε αυτή την ενότητα θα αναφερθούν επιγραμματικά οι παράμετροι που έχουν εξεταστεί κατά τη διεξαγωγή των πειραμάτων. Συγκεκριμένα θα εξεταστεί πώς συμπεριφέρεται ο αλγόριθμος: (α) Μεταβολή του Αριθμού των χρηστών (n) που χρησιμοποιούν την υπηρεσία, (β) False Positives

(α) Μεταβολή στον αριθμό των χρηστών (n)

Η μεταβολή στον αριθμό των χρηστών είναι ένας από τους σημαντικότερους παράγοντες που επηρεάζουν τη συμπεριφορά ενός οποιουδήποτε κατανεμημένου παράλληλου αλγόριθμου. Και σίγουρα δημιουργεί αρκετό ενδιαφέρον να απαντηθούν διάφορα ερωτήματα όπως, για τον συνολικό χρόνο που χρειάζεται να τρέξει ολόκληρο το πρόγραμμα, το συνολικό κόστος επεξεργασίας των δεδομένων και επίσης για το συνολικό κόστος επικοινωνίας των εξυπηρετητών που απαιτείται για να αλλάξουν μηνύματα μεταξύ τους.

Αναμένουμε ότι καθώς αυξάνεται ο αριθμός των χρηστών (n) αναλογικά αυξάνεται και ο συνολικός χρόνος που χρειάζεται για να τρέξει το πρόγραμμα και άρα αφού αυξάνονται οι χρήστες θα αυξάνονται και τα μηνύματα των χρηστών και άρα ως επακόλουθο θα αυξάνεται και ο χρόνος που απαιτείται για επεξεργασία των δεδομένων. Όμως το πλέον σημαντικό θα είναι το κόστος επικοινωνίας που θα έχουν οι εξυπηρετητές. Αναμένουμε ότι στη τελευταία προσέγγιση αυτό το κόστος θα παραμείνει σχετικά σταθερό και θα παίρνει μικρή τιμή λόγω της βελτιστοποίησης των MinHash Signatures που ανταλλάζονται μόνο οι χρήστες που είναι σημαντικοί και έχουν πάνω από κάποιο όριο ομοιότητας με τον εξυπηρετητή που θα ανταλλαγούν.

(β) False Positives

Σε αυτό το πείραμα θα εξακριβωθεί κυρίως για τη τελευταία προσέγγιση πόσο κοντά στη πραγματικότητα είναι οι συστάσεις που γίνονται για τους χρήστες. Λόγω της φύσεως της προσέγγισης στην οποία χρησιμοποιείται MinHashing Signatures υπάρχει το αντίτιμο ότι μπορεί να διαγνωστεί λανθασμένα ότι κάποιος χρήστης ή ακόμα και εξυπηρετητής έχει με κάποιο εξυπηρετητή αντίστοιχα λανθασμένους κοινούς χρήστες. Αυτό θα μας απασχολήσει πειραματικά στη επόμενο κεφάλαιο.

Κεφάλαιο 7

Πειραματική Αποτίμηση

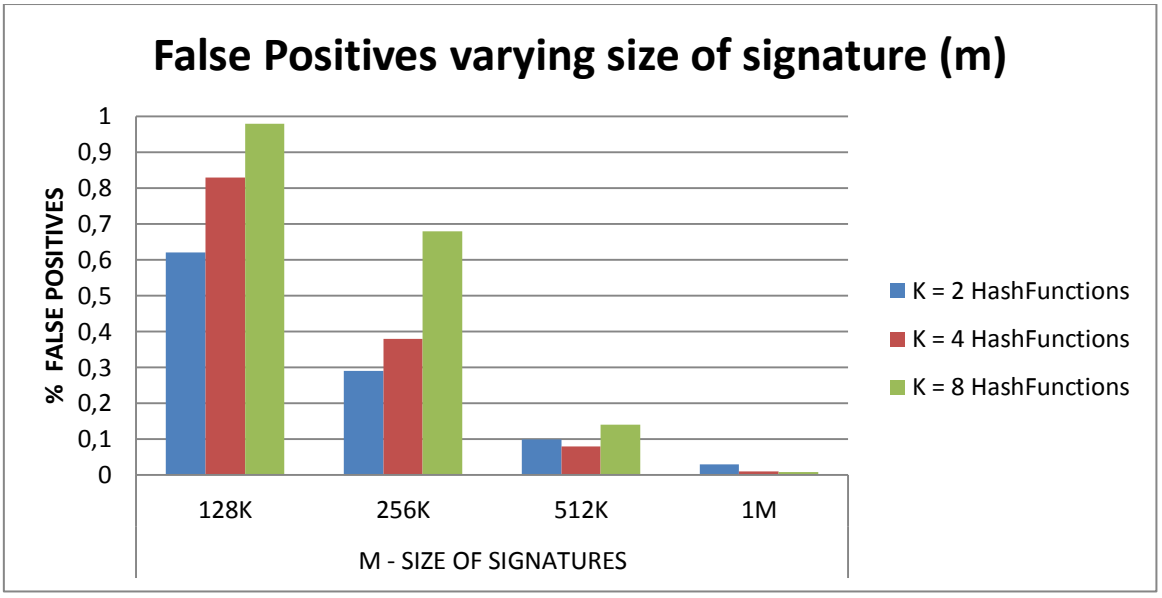
7.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει η πειραματική αποτίμηση των προσεγγίσεων που έχουν προταθεί για το ολοκληρωμένο σύστημα συστάσεων και θα γίνει επεξήγηση των γραφικών παραστάσεων που έχουν προκύψει μέσα από τα πειράματα που έχουν γίνει με τη μεταβολή του n που είναι ο αριθμός των χρηστών όπως επίσης και τη συμπεριφορά που έχει η προσέγγιση MinHashing Signatures σε σχέση με τα false positives τόσο στη φάση ανταλλαγής χρηστών στους εξυπηρετητές, όσο και στις τελικές συστάσεις.

7.2 Παρουσίαση και ανάλυση αποτελεσμάτων

Ερευνώντας την έννοια των false positives[21,22,23], οι έρευνες έχουν καταλήξει σε κάποια συμπεράσματα. Συγκεκριμένα όσων αφόρα την χρήση του BloomFilter που είναι μια ιδέα που έχει δώσει και τη παρακίνηση για τη δεύτερη προσέγγιση που έχει γίνει, ισχύει ένας μαθηματικός τύπος που μας δίνει κατά προσέγγιση τη πιθανότητα μέσα σε ένα σύνολο m από bits που έχει μια υπογραφή, σε σχέση με ένα σύνολο n από λέξεις μιας γλώσσας και εν τέλει με k συναρτήσεις κατακερματισμού, ποιο θα είναι το ποσοστό των false positives. Ο τύπος και η γραφική παράσταση για διάφορα αποτελέσματα παρουσιάζονται στο Σχήμα 7.1.

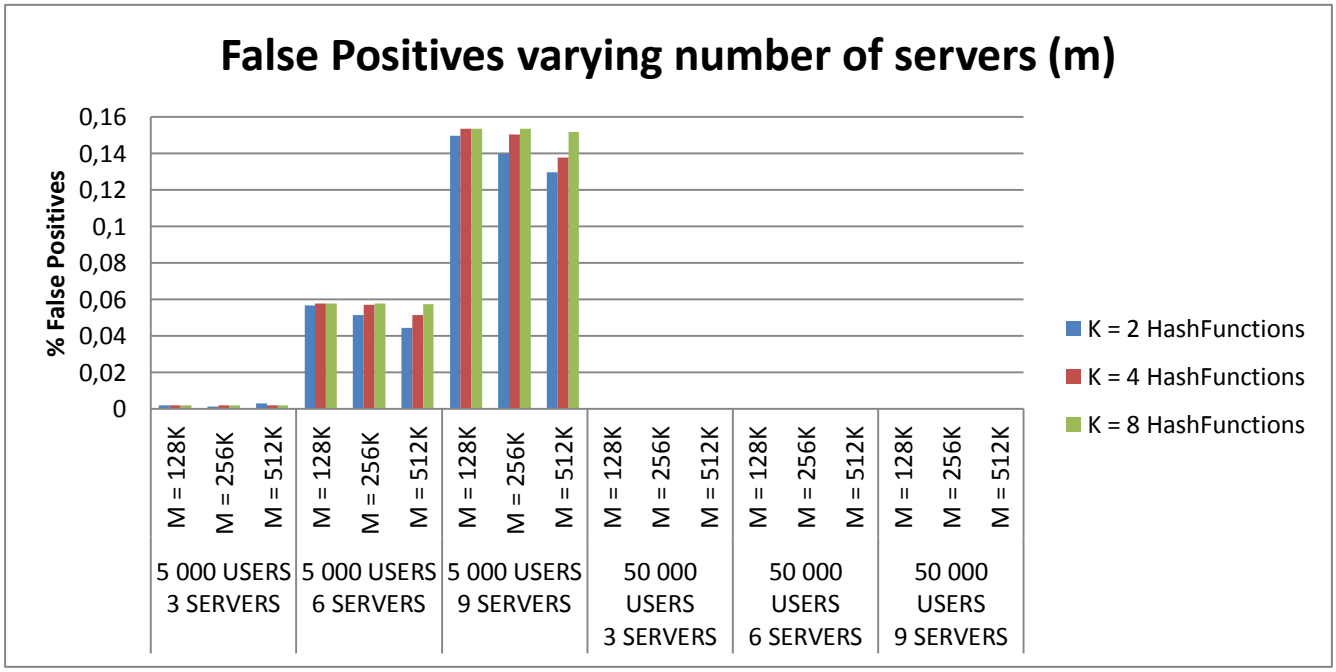
$$f_p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$$



Σχήμα 7.1 Κατανομή των false positives με N = 100,000 που είναι όλε οι λέξεις του λεξικού UNIX

Όπως βλέπουμε στη ποιο πάνω γραφική παράσταση με N = 100,000 λέξεις όπου οι λέξεις αυτές είναι όλες οι λέξεις του αγγλικού αλφαβήτου του λεξικού που παρέχεται στο UNIX σε μια υπογραφή που θα είναι μεγέθους M αν εφαρμόσουμε διαδοχικά σε όλες τις λέξεις του M, K συναρτήσεις κατακερματισμού βλέπουμε τα false positives. Πόσες θέσεις της υπογραφής είχαν ήδη τη τιμή 1 για κάθε λέξη που ελέγχουμε.

Θέλοντας να δούμε πειραματικά τη συμπεριφορά της δεύτερης προσέγγισης επιλέξαμε για M = 128K, 256K, 512K και αντίστοιχα για K = 2, 4 ,8 να τρέξουμε τα πειράματα για 5,000 και 50,000 χρήστες για να δούμε ποιο θα είναι το κατάλληλο μέγεθος για M και K το οποίο θα μας δίνει τα ποιο λίγα false positives στο κόστος επικοινωνίας. Σε αυτό το σημείο πρέπει να τονίσουμε ότι θεωρούμε ότι τα false positives υπάρχουν στο σημείο της ανταλλαγής των κοινών χρηστών. Εκεί θα πρέπει ο θόρυβος – noise που δημιουργείται να είναι λίγος, δηλαδή να μην έχουμε άσκοπη ανταλλαγή χρηστών γιατί το noise όπως έχουν καταδείξει και τα πειράματα που θα παρουσιαστούν στη συνέχεια μεταφράζεται ως αύξηση του κόστους επικοινωνίας των εξυπηρετητών (στέλνονται περισσότερα δεδομένα απ' ότι πρέπει), αύξηση του κόστους επεξεργασίας για την ανάκτηση του AkSNN, αλλά και απόκλιση στις σωστές συστάσεις σε σχέση με το ground truth που είναι η πρώτη προσέγγιση. Για να γίνει ο έλεγχος για τα false positives για τους 5,000 και 50,000 χρήστες έγινε σύγκριση των χρηστών που αποστέλλονται από κάθε εξυπηρετητή σε ένα άλλο για τις δύο προσεγγίσεις. Το τελικό αποτέλεσμα είναι ο μέσος όρος των false positives που προκύπτουν για κάθε μέγεθος χρηστών και παρουσιάζονται στο Σχήμα 7.2.

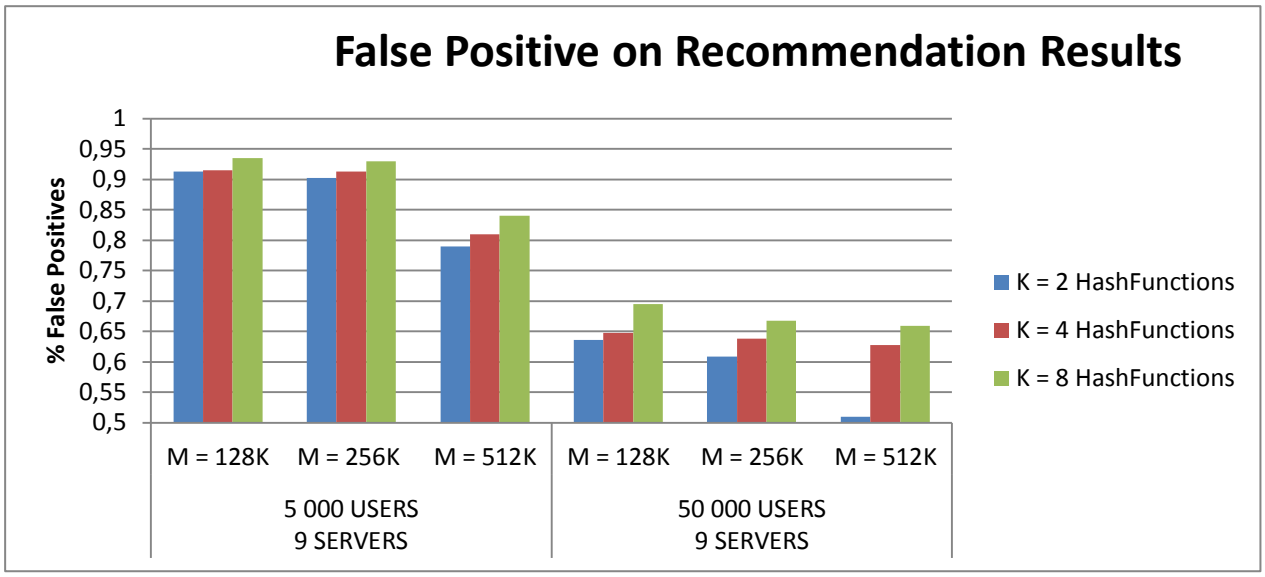


Σχήμα 7.2 Κατανομή των false positives με διαφορετικά M (Signature Size) και διαφορετικά μεγέθη χρηστών N

```
0 original tokens 25244 got 1 23861
0 interection with 1 = 7788 / 25244
0 original tokens 25244 got 2 26487
0 interection with 2 = 8454 / 25244
1 original tokens 23861 got 0 25244
0 original tokens 25244 got 3 23103
0 interection with 3 = 7541 / 25244
0 original tokens 25244 got 4 24186
0 interection with 4 = 7765 / 25244
1 interection with 0 = 7788 / 23861
0 original tokens 25244 got 5 24722
0 interection with 5 = 7969 / 25244
1 original tokens 23861 got 2 26487
1 interection with 2 = 7992 / 23861
1 original tokens 23861 got 3 23103
1 interection with 3 = 7021 / 23861
```

Σχήμα 7.3 Κοινά token για τη πρώτη προσέγγιση μεταξύ εξυπηρετητών

Η γραφική παράσταση στο Σχήμα 7.2 έχει προβληματίσει αρκετά την αξιολόγηση των πειραμάτων. Αρχικά για τους 5,000 φαίνεται ότι καθώς το πρόβλημα λύνεται με λιγότερους εξυπηρετητές τα false positives που προκύπτουν μεταξύ τους τείνουν να είναι ποιο λίγα σε σχέση με αυτά που προκύπτουν στους 9 εξυπηρετητές. Αυτό συμβαίνει διότι τα token σε μικρά dataset που κατανέμονται σε λίγους εξυπηρετητές, τείνουν να είναι μοναδικά ανάμεσα τους. Έτσι στους 3 εξυπηρετητές, τείνουν οι τομές τους να μην έχουν πολλά κοινά. Άρα δεν προκύπτει ιδιαίτερο κόστος επικοινωνίας το οποίο θα υποφέρει από false positives. Καθώς όμως το μέγεθος των εξυπηρετητών μεγαλώνει αυτό γίνεται ποιο έντονο.

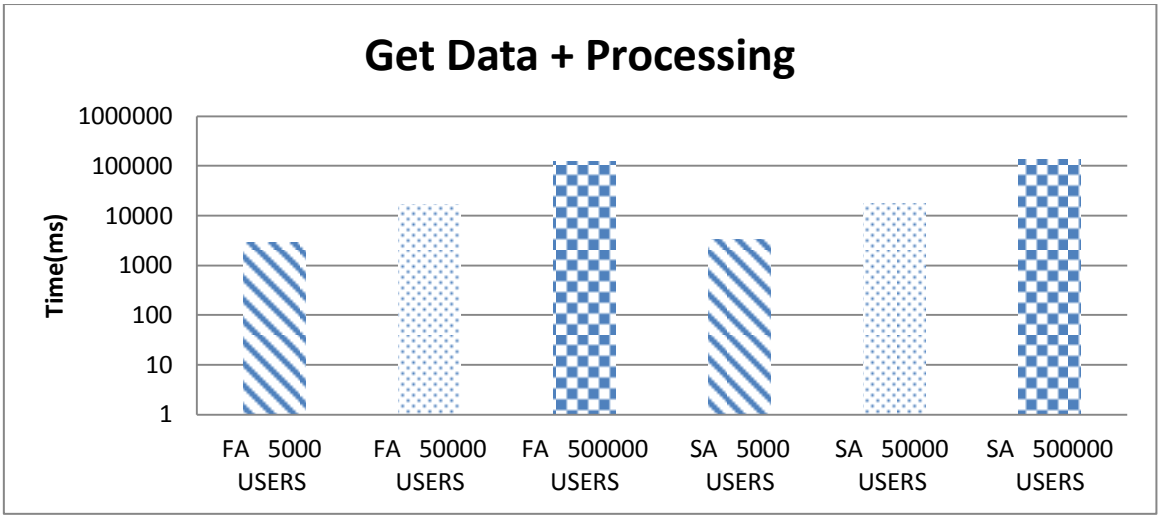


Σχήμα 7.4 False Positives στα τελικά αποτελέσματα συστάσεων

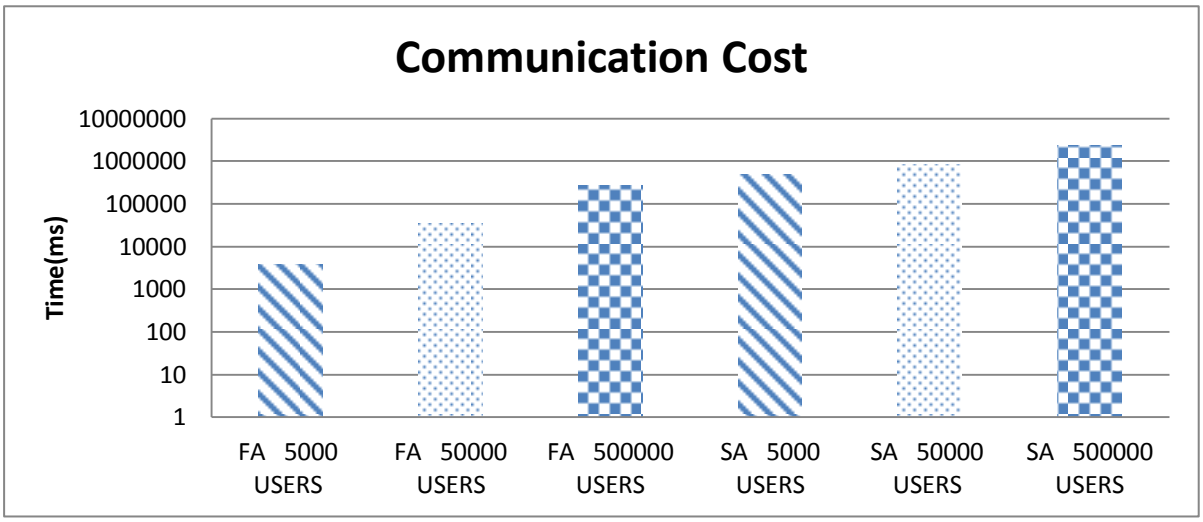
Αυτό που φαίνεται στους 50,000 χρήστες είναι ότι δεν υπάρχει false positives. Αυτό οδήγησε στο να γίνει ένα δεύτερο πείραμα που φαίνεται στο Σχήμα 7.3 όπου για τους 50,000 χρήστες τυπωνόταν η τομή των token (για τη προσέγγιση με τα Inverted Index), όπου με ένα άλλο πρόγραμμα υπολογιζόταν ο συνολικός αριθμός των χρηστών που θα γινόταν η ανταλλαγή μεταξύ των εξυπηρετητών. Ο αριθμός αυτός ήταν ίσος με το συνολικό αριθμό των χρηστών που ανήκαν σε κάθε εξυπηρετητή αντίστοιχα. Άρα ο κάθε εξυπηρετητής θα έστελνε σε όλους τους άλλους εξυπηρετητές, όλους του τους χρήστες. Γι' αυτό και στη σύγκριση της πρώτης με της δεύτερης προσέγγισης το ποσοστό των false positives που προέκυπτε για τη δεύτερη προσέγγιση ήταν 0.

Έτσι αφού έχει διαπιστωθεί αυτό το πρόβλημα, έγινε το επόμενο πείραμα που θα έδειχνε τα false positives, που προκύπτουν πάνω στις πραγματικές συστάσεις για να αξιολογηθεί με αυτό τον τρόπο πώς συμπεριφέρεται ο αλγόριθμος με τις υπογραφές σε σχέση με την προσέγγιση με τα Inverted Index. Τα αποτελέσματα φαίνονται στην γραφική παράσταση στο Σχήμα 7.4.

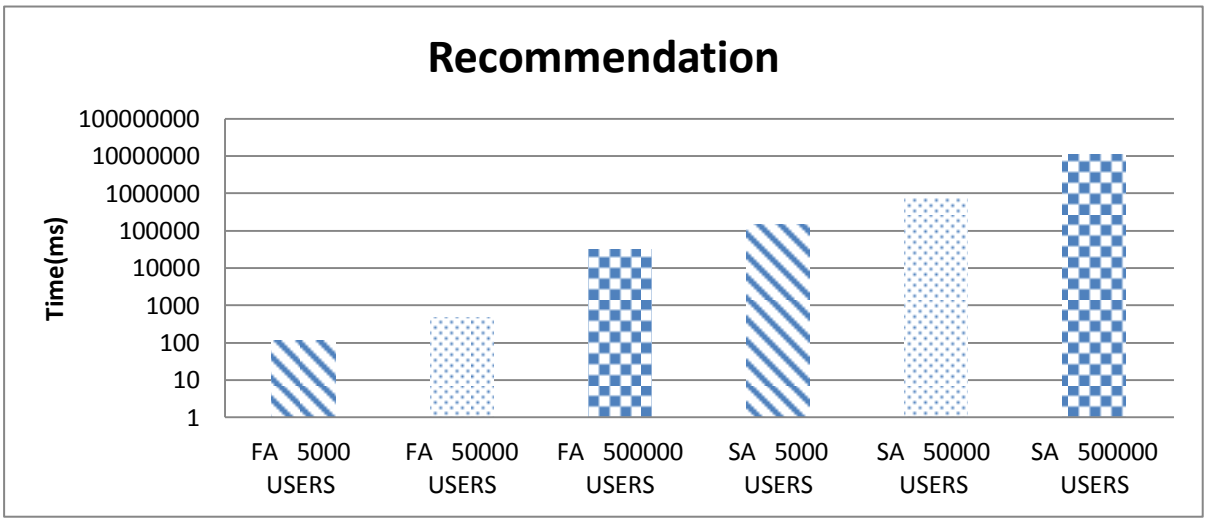
Σε αυτό το σημείο πρέπει να αναφερθεί το ότι έγινε η σκέψη για καλυτέρευση της δεύτερης προσέγγισης με τις υπογραφές. Συγκεκριμένα για να μειωθεί το μέγεθος της υπογραφής για κάθε χρήστη. Με αυτό τον τρόπο ο κάθε εξυπηρετητής θα στέλνει τις υπογραφές των χρηστών του στους άλλους εξυπηρετητές και μόνο όσοι χρήστες πρέπει να ανταλλαγούν θα ανταλλάσσονται. Όμως σύμφωνα με το πείραμα που έχει αναφερθεί προηγουμένως και με το Σχήμα 7.4 αυτό δεν θα μπορούσε να γίνει γιατί καθώς μειώνεται το μέγεθος υπογραφής αυξάνονται και τα false positives



Σχήμα 7.5 Κόστος Επεξεργασίας δεδομένων στις δύο προσεγγίσεις



Σχήμα 7.6 Κόστος Επικοινωνίας Εξυπηρετητών



Σχήμα 7.7 Συνολικό Κόστος Συστάσεων

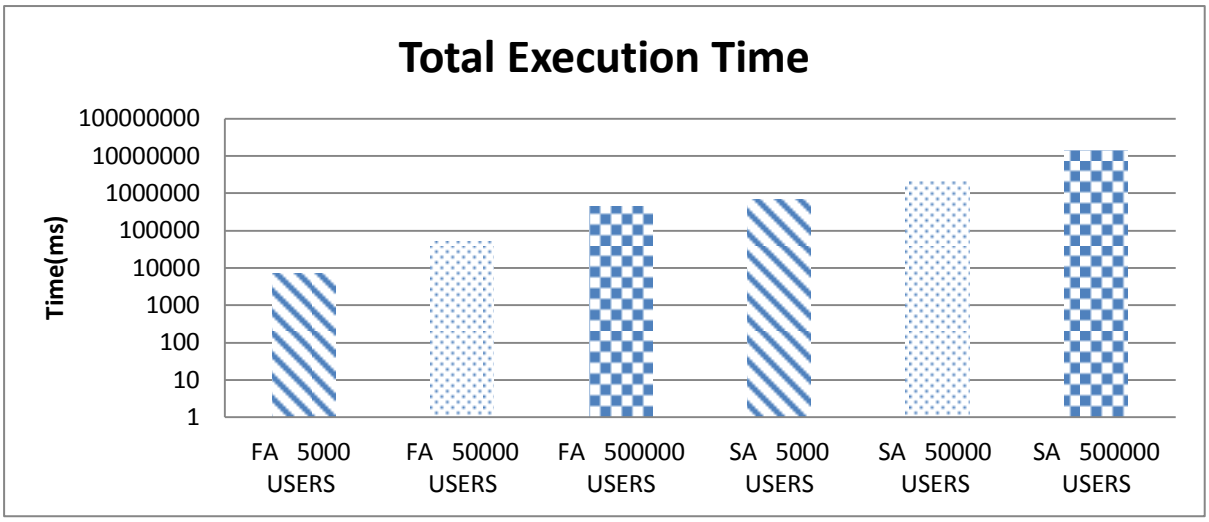
Στο Σχήμα 7.5 φαίνεται λογαριθμικά το κόστος επεξεργασίας των δεδομένων για τις δύο προσεγγίσεις για 9 εξυπηρετητές, μέγεθος υπογραφής 512K και 2 συναρτήσεις κατακερματισμού. Το κόστος κυμαίνεται στα ίδια επίπεδα για τις δύο προσεγγίσεις αφού

στην μεν πρώτη γίνεται η χρήση των πινάκων κατακερματισμού που μας επιτρέπουν γρήγορη εισαγωγή και αναζήτηση τόσο στα ενδιαφέροντα των χρηστών του εξυπηρετητή όσο και στο να βρίσκουμε τους χρήστες και να ενημερώνουμε τις δομές τους. Στη δεύτερη προσέγγιση από την άλλη πάλι δεν γίνεται κάτι εξαιρετικά πολύπλοκο όσων αφορά τις πράξεις αφού ουσιαστικά γίνεται ο κατακερματισμός των λέξεων K φορές μέσα στην υπογραφή του κάθε χρήστη.

Μια από τις σημαντικότερες γραφικές παραστάσεις στα πειράματα που έχουν γίνει για διαφορετικά μεγέθη χρηστών είναι αυτή του Σχήματος 7.6 που καταδεικνύει το κόστος επικοινωνίας των χρηστών. Όπως βλέπουμε το λογαριθμικό κόστος επικοινωνίας για τη πρώτη προσέγγιση με N = 500,000 είναι στο ίδιο επίπεδο με το κόστος επικοινωνίας με N = 5,000 χρήστες που ισχύει για τη δεύτερη προσέγγιση με τις υπογραφές. Αυτό ισχύει για την με την εξής παρατήρηση.

Οι χρήστες στην πρώτη προσέγγιση που ανταλλάζονται στους εξυπηρετητές κάθε φορά έχουν ένα σταθερό μέγεθος των περίπου 40 bytes. Αυτό δίνει ένα πλεονέκτημα στην πρώτη προσέγγιση αφού το κόστος επικοινωνίας για τους χρήστες είναι λογαριθμικά σχετικά μικρό αν συγκριθεί με αυτό της δεύτερης προσέγγισης. Στη δεύτερη προσέγγιση οι χρήστες ανταλλάζονται με τον εξής τρόπο. Ο κάθε χρήστης θα συγκριθεί με τον κάθε εξυπηρετητή και αν η τομή στις υπογραφές τους δείξει πολλαπλάσιο K κοινών άσσων τότε οι χρήστες αυτοί θα ανταλλαγούν. Η ανταλλαγή γίνεται με αυτό τον τρόπο γιατί θέλουμε ακριβώς να ανταλλαγούν οι σωστοί χρήστες που έχουν πολλαπλάσιο K κοινών ενδιαφερόντων σε σχέση με τους εξυπηρετητές. Ένα απλό παράδειγμα θα μπορούσε να είναι η λέξη «android». Αν αυτή η λέξη θα γυρίσει K θέσεις στην υπογραφή τότε αναμένουμε σε δύο υπογραφές που θα συγκριθούν να βρούμε K θέσεις κοινές που θα έδιναν το «android». Αυτό όμως δίνει αρκετά false positives όπως λέχθηκε και προηγουμένως Αυτό συνεπάγεται αμέσως αύξηση στο κόστος επικοινωνίας το οποίο στο Σχήμα 7.6 όπως θα επεξηγηθεί και στη συνέχεια θα σημαίνει αύξηση του κόστους για τις συστάσεις. Άρα όπως καταλήγουμε σε αυτό το σημείο το κόστος μεταφοράς των λανθασμένων χρηστών κάνει τη χρήση της δεύτερης προσέγγισης ασύμφορη από τη πρώτη γιατί ακριβώς το κόστος επικοινωνίας μεταφράζεται σε κόστος επεξεργασίας.

Στο Σχήμα 7.7 φαίνεται αυτό που έχει λεχθεί και ποιο πάνω. Επειδή αποστέλλονται περισσότεροι χρήστες αυτό δημιουργεί προβλήματα στις συστάσεις. Συγκεκριμένα όπως έχει καταδείξει και η πολυπλοκότητα για τη δεύτερη προσέγγιση σε προηγούμενο κεφάλαιο, ο κάθε χρήστης έχει ένα συγκεκριμένο μέγεθος. Έστω ότι στη δεύτερη προσέγγιση ο χρήστης έχει μέγεθος 512KB. Ο χρήστης πρέπει να αποσταλεί μαζί με την υπογραφή του στον άλλο



Σχήμα 7.6 Συνολικό Κόστος Εκτέλεσης

εξυπηρετητή. Δηλαδή για κάθε χρήστη της $2^{η}$ προσέγγισης αν το συγκρίνουμε με ένα χρήστη της πρώτης προσέγγισης που έχει μόνο 14 bytes (4 bytes για ένα νούμερο που δείχνει τον βάρος που έχει σε ένα token και το όνομα του, επί τα token που βρίσκεται μέσα όπου για κάθε χρήστη αν θεωρήσουμε ότι η κατανομή είναι τυχαία τότε περίπου ανέρχονται στα 30 (περίπου 5 rayz επί περίπου 5 token = 25 έστω ότι είναι 30)). Άρα ενώ για ένα χρήστη θα χρειαστούμε 14 * 30 = 420 bytes αν το συγκρίνουμε με 512KB τότε η πράξη 512,000/120 = 1219. Άρα με ένα χρήστη της $2^{ης}$ προσέγγισης στέλνουμε 1219 στη πρώτη!

Αυτό καταδεικνύει το πρόβλημα που υπάρχει στη $2^{η}$ προσέγγιση με την ανταλλαγή των χρηστών ιδίως όταν υπάρχει και αυξημένο, 10% false positives. Σε αυτό το σημείο όμως πρέπει να εξεταστεί και το κόστος των συστάσεων μετά την ανταλλαγή. Έστω ότι στο σημείο πριν την ανταλλαγή των χρηστών ο κάθε εξυπηρετητής έχει n_{s_i} χρήστες. Μόλις γίνει αυτή ανταλλαγή θα προκύψουν $n_{s_i} + \sum_{j=1}^{m} v_j \;\; i \neq j$. Το άθροισμα είναι όλοι οι χρήστες που ο εξυπηρετητής s_i θα λάβει από τον s_j. Άρα το κόστος εκτέλεσης του recommendation θα είναι $n_{s_i} * (n_{s_i} + \sum_{j=1}^{m} v_j \;\; i \neq j)$.Αυτό έχει μια γενική πολυπλοκότητα O (n^2). Θα πρέπει να ελέγξουμε όλους τους χρήστες που είχε πριν ένας εξυπηρετητής με όλους τους υπόλοιπους που έχουν συμπληρωθεί μετά την ανταλλαγή. Άρα θα πρέπει να εκτελέσουμε μια n^2 εκτέλεση για να βρούμε τις πραγματικές συστάσεις.

Στη πρώτη προσέγγιση το κόστος πολυπλοκότητας είναι $n_{s_i} * T_{u_i} * G_t$. Επειδή όμως η κατανομή είναι τυχαία μέσα στο σύνολο $T_{u_i} * G_t$ δεν θα υπάρχουν ποτέ όλοι οι χρήστες. Άρα η πολυπλοκότητα για τη πρώτη προσέγγιση δεν θα είναι ποτέ O (n^2) αλλά θα είναι O $(n_{s_i} * T_{u_i} * G_t)$. Με αυτή την εξήγηση καταλήγουμε στην τελευταία γραφική παράσταση και στο Σχήμα 8.6 στην οποία και δικαιολογείται το γιατί στη $2^{η}$ προσέγγιση υπάρχει σοβαρό πρόβλημα στο χρόνο αποπεράτωσης κάτι που την κάνει ασύμφορη προς χρήση.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντικές Επεκτάσεις

8.1 Συμπεράσματα

Στη προηγούμενο κεφάλαιο έχουν παρουσιαστεί τα αποτελέσματα που έχουν προκύψει για τις δυο προσεγγίσεις που έχουν υλοποιηθεί. Η μεν πρώτη προσέγγιση θεωρείται ως ground truth. Δηλαδή το αποτέλεσμα που μας δίνει μπορεί να συγκριθεί με οποιαδήποτε άλλη προσέγγιση. Είχε επιλεγεί η προσέγγιση με τις υπογραφές η οποία θεωρητικά θα έδινε παρόμοια αποτελέσματα σε λιγότερο χρόνο με κάποια false positives.

Η προσέγγιση με τις υπογραφές έχει στο σημείο που ανταλλάζονται χρήστες και στη φάση της σύστασης, false positives. Στην μεν πρώτη περίπτωση τα false positives που θα προκύψουν από το communication phase θα δημιουργήσουν noise μέσα στα δεδομένα τα οποία ανταλλάζονται και η επίδραση τους φαίνεται στις τελικές συστάσεις, δηλαδή στα false positives της φάσης σύστασης. Συνοπτικά καταλήγουμε ότι ανάλογα με τα false positives που υπάρχουν στην φάση ανταλλαγής των χρηστών, ανάλογα αυξάνεται και το κόστος μεταφοράς αυτών των χρηστών, όπου εν τέλει αυξάνεται και το τελικό κόστος σύστασης.

Αρχικά τα πειράματα έχουν καταδείξει ότι για μικρά dataset, δηλαδή για dataset που αφορούν λίγους χρήστες (μέχρι 5,000 χρηστών) τα false positives που προκύπτουν στη φάση ανταλλαγής χρηστών τείνουν να είναι λιγότερα όταν έχουμε λίγους εξυπηρετητές λόγω της μοναδικότητας της κατανομής των token σε αυτούς. Αντίθετα καθώς αυξάνεται ο αριθμός των εξυπηρετητών τα false positives που προκύπτουν σε αυτή τη περίπτωση αυξάνεται.

Για ποιο μεγάλα dataset (μεγαλύτερα των 5,000 χρηστών) παρατηρείται πλήρης ανταλλαγή των χρηστών λόγω του ότι τα token κατανέμονται στο χώρο καλύτερα ανάμεσα στους εξυπηρετητές. Έτσι όλοι οι εξυπηρετητές έχουν κοινά token, και στο άθροισμα τους βρίσκονται όλοι οι πιθανοί χρήστες που πρέπει να ανταλλαγούν σε όλους τους εξυπηρετητές. Αυτό συμβαίνει και στις δύο προσεγγίσεις.

Οπόταν η δεύτερη προσέγγιση καθίσταται ασύμφορη προς χρήση λόγω του μεγάλου κόστος επικοινωνίας που υπάρχει για τη μεταφορά των χρηστών ανάμεσα στους εξυπηρετητές κάτι που έρχεται σε αντίθεση με τη πρώτη προσέγγιση όπου το κόστος είναι μικρότερο και επίσης οι χρήστες μεταφέρονται ευρετηριασμένοι. Με αυτό καταλήγουμε στο ότι η δεύτερη προσέγγιση είναι ασύμφορη.

Αν όμως θεωρήσουμε ότι δεν θέλουμε το ground truth που δίνει η πρώτη προσέγγιση και θα δώσουμε σημασία στο ποσοστό ομοιότητας που έχουν μεταξύ τους ένας εξυπηρετητής με ένα άλλο αλλά και ένας χρήστη με ένα εξυπηρετητή τότε στη δεύτερη προσέγγιση οι χρήστες μπορούν να φιλτράρονται. Αυτό μπορεί να μειώσει αρκετά το χρόνο αποπεράτωσης αλλά τα αποτελέσματα δεν θα είναι σωστά όπως της πρώτης προσέγγισης.

Σαφώς καταλήγουμε στο συμπέρασμα ότι η $1^{η}$ προσέγγιση και η χρήση των Inverted Index δίνουν καλύτερα αποτελέσματα από τη $2^{η}$ προσέγγιση που γίνεται η χρήση MinHash Signatures όπου πειραματικά τα αποτελέσματα έχουν δείξει ότι για μισό εκατομμύριο χρήστες ο συνολικός χρόνος εκτέλεσης κυμαίνεται στα περίπου 3,5 λεπτά από τα οποία ο χρόνος για το κομμάτι που επιτελεί τη σύσταση είναι περίπου 1 λεπτό. Αυτό ενθαρρύνει την χρήση των Inverted Index σε συστήματα συστάσεων αφού σαν συνολικός χρόνος είναι αρκετά ικανοποιητικός. Ιδίως αν λάβουμε υπόψη ότι σε πολύ λίγο χρονικό διάστημα έχουμε σωστά αποτελέσματα με βάση τις πραγματικές προτιμήσεις των χρηστών αφού λαμβάνεται υπόψη το πραγματικό βάρος που έχει ένας χρήστης με τα ενδιαφέροντα του σε σχέση με την απόσταση που έχει με ένα χρήστη αφού τα διαφορετικά βάρη όπως έχει αναφερθεί και στο παρελθόν γίνονται normalize και έτσι δίνεται η αναλογία 50% τόσο στο βάρος όσο και στην απόσταση.

8.2 Μελλοντικές Επεκτάσεις

Η συγκεκριμένη διπλωματική εργασία μπορεί να αναπτυχθεί περεταίρω με διάφορους τρόπους. Όπως έχει λεχθεί και στο προηγούμενο κεφάλαιο η χρήση των MinHashing Signatures μπορεί να γίνει με κάπως διαφορετικό τρόπο αν δεν μας ενδιαφέρει το ground truth όπως επεξηγήθηκε και ποιο πάνω.

Μια άλλη πιθανή προέκταση της εν λόγω διπλωματικής εργασίας είναι να χρησιμοποιήσουμε τα αποτελέσματα που προκύπτουν από τις συστάσεις και να εκτελέσουμε ομαδοποίηση των χρηστών αναλόγως των συστάσεων. Με αυτό τον τρόπο θα έχουμε ομαδοποιημένους χρήστες όπου το σύστημα τους έκανε κοινές συστάσεις. Αυτούς τους χρήστες μπορούμε κάθε ένα συγκεκριμένο χρονικό διάστημα να τους αποστέλλεται ένα μήνυμα rayz και να τους ειδοποιεί ότι σύμφωνα με τις συστάσεις τους έχουν κοινά ενδιαφέροντα και να τους δίνει τα ενδιαφέροντα και έτσι με αυτό τον τρόπο να προάγεται η συζήτηση. Με αυτό τον τρόπο ίσως να δίνουμε ένα κίνητρο στους χρήστες να χρησιμοποιούν περισσότερο την υπηρεσία και να περιμένουν από το σύστημα αυτή την ευκαιρία να συνομιλήσουν σε πραγματικό χρόνο με άλλους χρήστες που μοιάζουν μεταξύ τους.

Τέλος θα πρέπει ο αλγόριθμος να ενσωματωθεί στην υπηρεσία Rayzit.

Βιβλιογραφία

[1] **Su Mon Kywe, Ee-Peng Lim and Feida Zhu.** A Survey of Recommender Systems in Twitter

[2] **Armentano, M.G., Godoy, D.L., Amandi.** Recommending Information Sources to Information Seekers in Twitter. In: International Workshop on Social Web Mining A Survey of Recommender Systems in Twitter 13

[3] **Armentano, M.G., Godoy, D.L., Amandi.** Towards a Followee Recommender System for Information Seeking Users in Twitter. In: The 2nd International Work- shop on Semantic Adaptive Social Web

[4] **Armentano, M.G., Godoy, D.L., Amandi.** A Topology-Based Approach for Followees Recommendation in Twitter. In: 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems. Barcelona, Spain (July 2011)

[5] **Garcia, R., Amatriain, X.:** Weighted Content Based Methods for Recommending Connections in Online Social Networks. In: The 2nd ACM Workshop on Recommendation Systems and the Social Web. Barcelona, Spain (June 2010)

[6] **Golder, S.A., Marwick, A., Yardi, S., Boyd, D.:** A structural approach to contact recommendations in online social networks. In: Workshop on Search in Social Media, In conjunction with ACM SIGIR Conference on Information Retrieval

[7] **Hannon, J., Bennett, M., Smyth, B.:** Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches. In: The 4th ACM Conference on Recommender Systems (2010).

[8] **Amazon.com Recommendations:** Item-to-Item Collaborative Filtering. Industry Report

[9] **Xiaohui Yu, Ken Q. Pu, Nick Koudas:** Monitoring k-Nearest Neighbor Queries Over Moving Objects

[10] **Yunjun Gao, Baihua Zheng, Gencai Chen, Qing Li , Xiaofa Guo:** Continuous visible nearest neighbor query processing in spatial databases.

[11] **Georgios Chatzimilioudis, Demetrios Zeinalipour-Yazti , Wang-Chien Lee, Marios D. Dikaiakos :** Continuous All k-Nearest Neighbor Querying in Smartphone Networks

[12] **Constantinos, C.(2012).** Scalable continuous all k nearest neighbors queries, Published master dissertation, University of Cyprus, Nicosia, Cyprus.

[13] **Justin Zobel and Alistair Moffat:** Inverted Files for Text Search Engines

[14] **Hao Yan, Shuai Ding, Torsten Suel**: Inverted Index Compression and Query Processing with Optimized Document Ordering

[15] **Daniele Broccolo, Lorenzo Marcon, Franco Maria Nardini, Raffaele Perego, Fabrizio Silvestri:** Generating suggestions for queries in the long tail with an inverted index

[16] **Daniele Broccolo, Lorenzo Marcon , Franco Maria Nardini, Raffaele Perego, Fabrizio Silvestri :** Generating Suggestions for Queries in the Long Tail with an Inverted Index

[17] **Anand Rajaraman, Jure Leskovec, Jeffrey D. Ullman:** Mining of Massive Datasets

[18] **Yunyao Li, Vivian Chu, Sebastian Blohm :** Facilitating Pattern Discovery for Relation Extraction with Semantic-Signature-based Clustering

[19] **Dimo Brockhoff ,Eckart Zitzler:** Objective Reduction in Evolutionary Multiobjective Optimization: Theory and Applications

[20] **Andreas Konstantinidis, Christos Aplitsiotis and Demetrios Zeinalipour-Yazti: SmartP2P:** A Multiobjective Framework for Finding Social Content in P2P Smartphone Networks

[21] **Yury Lifshits, Shengyu Zhang,** Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design

[22] **Ken Christensen, Allen Roginsky, Miguel Jimeno:** A new analysis of the false positive rate of a Bloom filter

[23] **Benoit Donnet, Bruno Baynat, Timur Friedman:** Improving retouched Bloom filter for trading off selected false positives against false negatives