Χαροκόπειο Πανεπιστήμιο

Πληροφορικής και
Τηλεματικής

Πτυχιακή Εργασία

Βελτίωση Συστημάτων Συστάσεων (recommender systems) με αξιοποίηση Κοινοτήτων Χρηστών

Στρούμπος Αριστείδης

Επιβλέπων Καθηγητης:
Βαρλάμης Ηρακλής

Μέλη της Εξεταστικής Επιτροπής:
Μιχαήλ Δημήτρης
Κωνσταντίνα Βασιλοπούλου

Αθήνα,
Φεβρουάριος 2014

Περίληψη

Βελτίωση Συστημάτων Συστάσεων (recommender systems) με αξιοποίηση Κοινοτήτων Χρηστών

Η παρούσα έρευνα περιλαμβάνει την μελέτη της θεωρίας και των μεθόδων που χρειάζονται για την δημιουργία ενός συστήματος συστάσεων το οποίο εκμεταλλεύεται τις πληροφορίες που αντλεί από ένα κοινωνικό γράφο. Επίσης περιγράφεται ο σχεδιασμός μιας πλατφόρμας εύρεσης κοινοτήτων χρηστών που θα χρησιμοποιηθεί ως εργαλείο για την επίτευξη των παραπάνω. Αρχικά σε ένα πρώτο κομμάτι της έρευνας θα αναλυθούν οι βασικές έννοιες και προσεγγίσεις των Συστημάτων Συστάσεων σύμφωνα με την σχετική βιβλιογραφία. Στην συνέχεια μελετάται πως ένα τέτοιο σύστημα μπορεί να βελτιωθεί χρησιμοποιώντας την επιστήμη των κοινωνικών δικτύων. Ακολουθεί η παρουσίαση της δικής μας υλοποίησης ενός τέτοιου συστήματος και τέλος περιγράφεται η ενσωμάτωση του προγράμματός μας σε ένα υπάρχον σύστημα προσωποποίησης ώστε να επεκταθεί η λειτουργία του και να βελτιωθεί η ακρίβεια των αποτελεσμάτων του.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συστήματα Συστάσεων βασιζόμενα σε Κοινωνικά Δίκτυα (SNRS)

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Προσωποποίηση, Συστήματα Συστάσεων, Κοινωνικά Δίκτυα, Κοινοτικές Δομές Δικτύων, Εξόρυξη Δεδομένων από Κοινωνικούς Γράφους

Abstract

Improving Recommender Systems by utilization of User Communities

This research includes the study of theory and methods needed to create a Recommender System which exploits the information derived from a social graph. Also we outline the design of a platform which locates User Communities that will be used as a tool for achieving the above. The first part of the research analyzes the basic concepts and approaches of Recommender Systems in accordance with the relevant literature. Furthermore we make a study of how such a system can be improved if combined with the field of social networks. Next is the presentation of our own implementation of such a system and finally we describe the integration of our program in an existing personalization system to extend function and improve the accuracy of its results.

SUBJECT AREA: Social Network based Recommender System (SNRS)

KEYWORDS: Personalization, Recommender Systems, Social Network, Community Structure in Networks, Mining Data from Social Graphs

Ευχαριστίες

Η εκπόνηση της παρούσας πτυχιακής εργασίας ξεκίνησε στο Τμήμα Πληροφορικής και Τηλεματικής του Χαροκοπείου Πανεπιστημίου Αθηνών τον Οκτώβριο του 2013 και ολοκληρώθηκε τον Φεβρουάριο του 2014. Σε αυτό το σημείο, θα ήθελα να εκφράσω τις ευχαριστίες μου σε όλους όσους συνέβαλαν άμεσα ή έμμεσα στην ολοκλήρωση της πτυχιακής εργασίας μου και γενικότερα των προπτυχιακών σπουδών μου.

Αρχικά θα ήθελα να εκφράσω ιδιαίτερες ευχαριστίες στον επιβλέποντα καθηγητή, Ηρακλή Βαρλάμη, για την καθοδήγηση, την ενθάρρυνση αλλά και την εμπιστοσύνη που μου έδειξε αναθέτοντας μου την συγκεκριμένη εργασία.

Στη συνέχεια, θα ήθελα να ευχαριστήσω την SciFy και τα μέλη της, ιδιαίτερα τον Βασίλη Σαλαπάτα και Γιωργο Γιαννακοπουλο των οποίων οι γνώσεις, οι συμβουλές και η στήριξη ήταν ζωτικής σημασίας για την ολοκλήρωση της πτυχιακής εργασίας.

Τέλος, ευχαριστώ το κοντινό οικογενειακό και φιλικό μου περιβάλλον, Χρήστο, Ευγενία, Αγγελική, Morgan, Παύλο, Λουκά, Γιάννη, Ντούμα, Εβίτα, για την καθημερινή κατανόηση, ψυχολογική στήριξη και υπομονή.

Περιεχόμενα

Περίληψη **i**

Abstract **ii**

Ευχαριστίες **iii**

Περιεχόμενα **iv**

1 Εισαγωγή **1**
1.1 Ανάλυση Του Προβλήματος 1
1.2 Προσέγγιση 2

2 Θεωρητική Ανάλυση και Αλγόριθμοι **4**
2.1 Προσωποποίηση 4
2.2 Συστήματα Συστάσεων 4
2.2.1 Εισαγωγή Στα Συστήματα Συστάσεων 5
2.2.2 Μέθοδοι Συστάσεων 6
2.3 Κοινωνικά Συστήματα Συστάσεων 7
2.3.1 Κοινωνικά Δίκτυα και Συστήματα Συστάσεων 7
2.3.2 Συστάσεις βάση Εμπιστοσύνης και Κοινωνικές Συστάσεις 8
2.3.3 Πώς Θα Πετύχουμε Κοινωνικές Συστάσεις 9
2.3.4 Τι είναι και τι αντιπροσωπεύει μια Κοινότητα Χρηστών . 9
2.4 Αλγόριθμοι για την παραγωγή Κοινοτήτων Χρηστών 10
2.5 Ανάλυση των Αλγορίθμων 11
2.5.1 WeakComponent 12
2.5.2 EdgeBetweenness 13
2.5.3 BronKerbosch 14
2.5.4 Metis 15

3 Σχεδιασμός **17**
3.1 socialPServer 17
3.2 Κοινωνική Πληροφορία 19
3.3 Κοινότητες χρηστών 20
3.4 Προφίλ Χρήστη (userProfile) 20
3.5 PServer 20
3.6 Αρχιτεκτονική του PServer 22
3.7 ενσωμάτωση στον PServer 24

4 Υλοποίηση **26**
4.1 Πακέτο socialpserver 26
4.1.1 SocialPServer-main 27
4.1.2 Community 28
4.1.3 SetOfCommunities 29
4.2 Πακέτο socialpserver.algorithmic 32
4.2.1 interface ClustererAlgorithm 33
4.2.2 Ενδεικτικές Υλοποιήσεις Αλγορίθμων 34
4.3 Πακέτο socialpserver.dataio 38
4.3.1 Ανάκτηση - Αποθήκευση 39
4.3.2 DBAccess 40
4.3.3 Loggers 41

5 Αξιολόγηση παραγόμενων κοινοτήτων **43**
5.1 Δεδομένα (DataSets) 44
5.2 Σχεδιασμός της μεθόδου Αξιολόγησης 45
5.2.1 Αντιπροσωπευτικό σημείο συστάδας - centroid 46
5.2.2 Μετρική ομοιότητας - cosine similarity 47
5.2.3 Μετρική ομοιότητας ζεύγους συστάδων - Intra Simmilarity 47
5.2.4 Inter Simmilarity 48
5.3 Υλοποίηση της Αξιολόγησης 48

6 Εφαρμογή Προγράμματος και Αλγορίθμων Σε Πραγματικά Δεδομένα - Συμπεράσματα **50**
6.1 Εφαρμογή Σε Πραγματικά Δεδομένα 50
6.1.1 Χαρακτηριστικά των Αλγορίθμων: 51
6.1.2 Αλγόριθμοι Ομαδοποίησης (Clustering) 54
6.1.3 Αλγόριθμοι Κατάτμησης (Partitioning): Metis 56
6.2 Συμπεράσματα 58

7 Μελοντικές Επεκτάσεις - Προσωπικά Δεδομένα - Άδειες **59**
7.1 Μελλοντικές Επεκτάσεις 59
7.2 Προσωπικά Δεδομένα 60
7.3 Άδειες 61

Βιβλιογραφία **62**

Κεφάλαιο 1

Εισαγωγή

Η συνεχόμενη αύξηση του μεγέθους και της χρήσης του διαδικτύου έχει ως επακόλουθο οι υπηρεσίες να υπερφορτώνονται με πληροφορίες και η χρήση τους να γίνεται όλο και πιο πολύπλοκη. Οι χρήστες καθημερινά καλούνται να αντιμετωπίσουν την υπερπληροφόρηση η οποία συχνά μάλιστα τους εμποδίζει από το να εκπληρώσουν τον αρχικό τους σκοπό. Επομένως γίνεται όλο και πιο επιτακτική η ανάγκη για *προσωποποίηση των υπηρεσιών*. Προσωποποίηση θεωρείται η δυνατότητα μιας υπηρεσίας να λαμβάνει υπόψιν τις προτιμήσεις του κάθε χρήστη κατά την διαμόρφωση του περιεχομένου της.

1.1 Ανάλυση Του Προβλήματος

Η παρούσα εργασία ξεκινά με μια βασική υπόθεση: Έχουμε στη διάθεσή μας μια *γενικευμένη μηχανή προσωποποίησης περιεχομένου (PServer 3.5)* . Πρόκειται για έναν *εξυπηρετητή* ο οποίος όπως και κάθε μηχανή που προσφέρει προσωποποίηση, λαμβάνει υπόψιν τις προτιμήσεις του κάθε χρήστη και ακολούθως διαμορφώνει το περιεχόμενό της σύμφωνα με τα χαρακτηριστικά του κάθε ενός. Στην περίπτωση ενός δικτυακού τόπου ο PServer είναι ένας ακόμα *εξυπηρετητής* ο οποίος αναλαμβάνει να προσωποποιήσει οποιαδήποτε υπάρχουσα υπηρεσία, συλλέγοντας πληροφορίες για τους χρήστες και την αλληλεπίδραση τους με αυτήν και τελικά παρέχοντας τις πληροφορίες που χρειάζονται για την προσωποποίηση. Η διαδικασία αυτή για να πετύχει τους στόχους της εμπεριέχει ένα πλήρες *Σύστημα Συστάσεων*. Για την λειτουργία του, διατηρεί προφίλ χρηστών τα οποία τροφοδοτεί με δημογραφικές πληροφορίες και πληροφορίες προτιμήσεων και στη συνέχεια μπορεί να

προβλέψει: από όλο το εύρος των πιθανών προτιμήσεων ποιο είναι το υποσύνολο στο οποίο έγκειται ο κάθε χρήστης. Τελικώς, μπορεί να κάνει συστάσεις αντικειμένων στους χρήστες βασιζόμενος στην υπάρχουσα αλληλεπίδρασή τους με το σύστημα.

Καλούμαστε να επεκτείνουμε αυτό το σύστημα ώστε να δημιουργεί συστάσεις βασιζόμενο, όχι μόνο στις πληροφορίες των δημογραφικών στοιχείων και των προτιμήσεων, αλλά να μπορεί να προβλέπει την συμπεριφορά του κάθε χρήστη μελετώντας την κοινωνική δικτύωση των χρηστών της εφαρμογής. Με τον τρόπο αυτό αναμένουμε από το σύστημα να πριμοδοτεί περιεχόμενα με τα οποία είχαν επαφή οι φίλοι κάποιου χρήστη έναντι περιεχομένων που έχουν δει μόνο άγνωστοι σε αυτόν χρήστες. Για την εφαρμογή των παραπάνω, σχεδιάστηκε και υλοποιήθηκε ένα σύστημα (**socialPServer**) το οποίο επιτρέπει τον εντοπισμό των ενδιαφερόντων ενός χρήστη εξάγοντας συμπεράσματα από τα αντικείμενα τα οποία προτιμούν οι κοντινότεροι φίλοι του, δοσμένου του κοινωνικού γράφου της εφαρμογής. Το σύστημα αυτό μπορεί να χαρακτηριστεί ως **Social Recommendation System**.

1.2 Προσέγγιση

Τα *κοινωνικά δίκτυα* από τη φύση τους αποτελούν πολύ σημαντική πηγή εξόρυξης δεδομένων, καθώς η δομή τους εμπεριέχει πληθώρα πληροφοριών για τα μελή τους. Όταν μια εφαρμογή διατηρεί τον κοινωνικό γράφο των χρηστών της, μπορεί να εξάγει πληροφορίες οι οποίες ξεφεύγουν από τα πλαίσια της υπηρεσίας και αποκαλύπτουν στοιχεία για την πραγματική ζωή και συμπεριφορά των χρηστών. Αυτό προκύπτει από το γεγονός ότι ο άνθρωπος, ο οποίος από την φύση του είναι κοινωνικό όν, σε όλη την διάρκεια της ζωής του τείνει να ετεροκαθορίζεται και κατ' επέκταση να δρα επηρεασμένος από το κοινωνικό του περιβάλλον. Έτσι ο τομέας του *Social Analysis* αποκτά πολύ μεγάλη σημασία για την έρευνα αλλά και τις σημερινές ηλεκτρονικές υπηρεσίες. Πλέον η εξαγωγή γνώσης δεν εξαρτάται μόνο από την αλληλεπίδραση του χρήστη με το σύστημα, αφού ο κάθε χρήστης δεν αντιμετωπίζεται σαν μονάδα αποκομμένος από το σύνολο αλλά σαν μέλος του. Η επίδραση που έχει η κοινότητα στο κάθε μέλος βεβαίως είναι αμφίδρομη διαδικασία αφού όπως ο κάθε χρήστης επηρεάζεται από το γύρο περιβάλλον του, έτσι και ολόκληρο το κοινωνικό σύνολο διαμορφώνεται τελικά από τον κάθε ένα αυτόνομο χρήστη.

Για την πραγματοποίηση των παραπάνω, οι χρήστες της εφαρμογής ομαδοποιούνται σε κοινότητες χρηστών των οποίων τα μέλη έχουν μεταξύ τους ισχυρούς δεσμούς φιλίας. Έτσι σε επόμενο χρόνο μπορούν να σκιαγραφηθούν οι προτιμήσεις του κάθε χρήστη μέσω της ανάλυσης των συμπεριφορών και των προτιμήσεων των μελών της κοινότητας στην οποία ανήκει. Μέσω αυτής της διαδικασίας ο pServer θα είναι πλέον σε θέση να κάνει συστάσεις σε κάποιο χρήστη βασιζόμενος στις προτιμήσεις των κοντινότερων φίλων του από τους οποίους, όπως και στην πραγματική ζωή, εκτιμάται ότι δέχεται και την υψηλότερη επιρροή.

Κεφάλαιο 2

Θεωρητική Ανάλυση και Αλγόριθμοι

2.1 Προσωποποίηση

Η βασική φιλοσοφία ενός συστήματος Προσωποποίησης είναι *να παρέχει στους χρήστες τις πληροφορίες που θέλουν ή χρειάζονται, χωρίς να περιμένει να του ζητηθεί ρητά*.[1]
Τα δομικά στοιχεία της προσωποποίησης των δικτύων είναι:
- Η κατηγοριοποίηση και προεπεξεργασία των δεδομένων.
- Η εξόρυξη των συσχετίσεων μεταξύ διαφορετικών ειδών δεδομένων.
- Ο προσδιορισμός των ενεργειών που θα προταθούν από το σύστημα.[2]
Ο πιο διαδεδομένος τρόπος επίτευξης της προσωποποίησης είναι με την εφαρμογή ενός Συστήματος Συστάσεων. [3]

2.2 Συστήματα Συστάσεων

Ένα *Σύστημα Συστάσεων (Recommender System)* είναι ουσιαστικά ένας μηχανισμός που ως πρώτο σκοπό έχει το να προβλέψει το ενδιαφέρον ενός χρήστη για αντικείμενα με τα οποία δεν έχει ακόμα επαφή. Εντοπίζοντας και εμφανίζοντας στο χρήστη αντικείμενα με μέγιστο εκτιμώμενο ενδιαφέρον, επιτυγχάνεται η προσωποποίηση μιας Web υπηρεσίας και ο χρήστης καταφέρνει να αποφύγει την υπερπληροφόρηση έχοντας πρόσβαση σε ένα υποσύνολο των αντικειμένων της υπηρεσίας το οποίο τον ενδιαφέρει περισσότερο σύμφωνα με τις μέχρι στιγμής προτιμήσεις του. Σε αυτήν την παράγραφο

θα επιχειρήσουμε μία εισαγωγή στα Recommender Systems (συστήματα συστάσεων) και στις υπάρχουσες θεωρητικές και πρακτικές κατευθύνσεις.

2.2.1 Εισαγωγή Στα Συστήματα Συστάσεων

Στη μελέτη επισκόπησης των *Adomavicius* και *Tuzhilin* [4] όπου πραγματοποιήθηκε έρευνα σχετικά με το υπάρχον στάδιο των Recommender systems, αντλήθηκαν και παρουσιάζονται τα παρακάτω στοιχεία.

Τα συστήματα συστάσεων αναδύθηκαν σαν ερευνητικό πεδίο μετά την εμφάνιση των πρώτων papers σχετικών με το collaborative filtering στα μέσα της δεκαετίας του 90. Τόσο στην ακαδημαϊκή κοινότητα αλλά και στον επιχειρηματικό κόσμο, την τελευταία δεκαετία, γίνονται προσπάθειες εύρεση και υλοποίηση νέων προσεγγίσεων. Το ενδιαφέρον παραμένει ακόμα υψηλό καθώς υπάρχουν πολλά ακόμα προβλήματα προς επίλυση και χώρος για την ανάπτυξη προσωποποιημένων εφαρμογών, καθώς οι χρήστες των διαδικτυακών υπηρεσιών καλούνται να αντιμετωπίσουν την ολοένα και μεγαλύτερη υπερπληροφόρηση.

Σχετικές έννοιες είχαν ήδη αναφερθεί σε διαφορετικούς τομείς όπως στην ανάκτηση πληροφορίας (information retrieval) αλλά και στο μάρκετινγκ, αποτέλεσε όμως ανεξάρτητο πεδίο από την στιγμή που οι ερευνητές άρχισαν να μελετούν *προβλήματα βαθμονόμησης (ranking problems)* των συστάσεων. Ο πυρήνας ενός Συστήματος Συστάσεων είναι ουσιαστικά μια διαδικασία πρόβλεψης αξιολογήσεων(rating) των αντικειμένων με τα οποία ο χρήστης δεν είχε ακόμα επαφή, και στη συνέχεια βαθμονόμησης των αντικειμένων με βάση την αξιολόγηση, ώστε τελικά να προταθούν στον χρήστη τα αντικείμενα με την υψηλότερη εκτιμώμενη αξιολόγηση.

Αν επιχειρήσουμε μια πιο τυπική διατύπωση των παραπάνω έχουμε:
Ένα σύνολο χρηστών C και ένα σύνολο αντικειμένων S.
Επίσης έχουμε u τη συνάρτηση χρησιμότητας ενός αντικειμένου s για τον χρήστη c. Τότε για τη συνάρτηση χρησιμότητας u ισχύει ότι

$u : C$x$S \rightarrow R$, με R να είναι ένα καθορισμένο σύνολο (π.χ. μη αρνητικοί ακέραιοι ή πραγματικοί αριθμοί κάποιου εύρους).

Στην συνέχεια για κάθε χρήστη $c \in C$, θέλουμε να επιλέξουμε κάποιο αντικείμενο $\acute{s} \in S$ το οποίο να μεγιστοποιεί το εκτιμώμενο ενδιαφέρον (*χρησιμότητα*) του χρήστη για αυτό.

Επομένως: $\forall c \in C, \acute{s_c} = \arg_{s \in S} \max u(c, s)$

Συνήθως στα συστήματα συστάσεων η χρησιμότητα ενός αντικειμένου εκφράζεται με μια αξιολόγηση, εναλλακτικά η χρησιμότητα μπορεί να προκύπτει από μια καθορισμένη συνάρτηση κέρδους. Ανάλογα με την φύση της εφαρμογής, η χρησιμότητα u μπορεί είτε να δηλωθεί από τον χρήστη με την μορφή αξιολόγησης, είτε να υπολογιστεί από την ίδια την εφαρμογή όπως στην περίπτωση της συνάρτησης κέρδους.

Κάθε μέλος του συνόλου χρηστών C περιγράφεται από ένα *προφίλ (userProfile)* που περιλαμβάνει προσωπικές πληροφορίες, όπως δημογραφικά στοιχεία ή προτιμήσεις. Στην πιο απλή περίπτωση μοναδικό στοιχείο που χρειάζεται είναι ένα userID. Παρομοίως κάθε στοιχείο του συνόλου των αντικειμένων S μπορεί να περιγράφεται από ένα σύνολο χαρακτηριστικών. [4]

2.2.2 Μέθοδοι Συστάσεων

Hill, Resnick και Shardanand είναι μερικοί από τους θεμελιωτές της σύνθεσης κοινώς αποδεκτών μεθόδων για την παραγωγή συστάσεων. [5] [6] [7] Έκτοτε το πρόβλημα έλαβε τις υπάρχουσες διαστάσεις του και άρχισε να μελετάται περαιτέρω.

Τα Συστήματα Συστάσεων υπάγονται στις ακόλουθες γενικές κατηγορίες ανάλογα με τον τρόπο προσέγγισης των αξιολογήσεων:

Content-based recommendations
: Η πρόβλεψη των αξιολογήσεων προκύπτει από τις αξιολογήσεις που έχει κάνει ο χρήσης σε άλλα αντικείμενα. Επομένως τελικά προτείνονται αντικείμενα παρόμοια με αυτά που έχει αξιολογήσει θετικά στο παρελθόν.

Collaborative recommendations
: Η πρόβλεψη των αξιολογήσεων προκύπτει από τις αξιολογήσεις που έχουν κάνει άλλοι χρήστες της εφαρμογής, των οποίων τα προφίλ μοιάζουν με το προφίλ του χρήστη. Επομένως τελικά προτείνονται αντικείμενα τα οποία έχουν προτιμηθεί στο παρελθόν από χρήστες με κοινές προτιμήσεις ή χαρακτηριστικά.

Hybrid approaches
: Συνδυασμός των δυο παραπάνω μεθόδων.

Σε όλες τις περιπτώσεις όταν τα άγνωστα ratings έχουν πλέον υπολογιστεί συνίστανται στον χρήστη τα αντικείμενα με την υψηλότερη προβλεπόμενη αξιολόγηση.

Στα πλαίσια αυτής της πτυχιακής επιθυμούμε να εξάγουμε συμπεράσματα εκμεταλλευόμενοι τις προτιμήσεις άλλων χρηστών δηλαδή να δημιουργήσουμε συνεργατικές προτάσεις (Collaborative recommendations) αλλά με διαφορετικό τρόπο σε σχέση με τις παραδοσιακές μεθόδους όπως το Collaborative Filtering. Πρόκειται να πραγματοποιηθούν συστάσεις εξάγοντας γνώση όχι από την ομοιότητά των χρηστών αλλά από την κοινωνική τους διασύνδεση.

2.3 Κοινωνικά Συστήματα Συστάσεων

Τίθεται λοιπόν το ερώτημα: ”Ποια η σχέση των κοινωνικών δικτύων με τα συστήματα συστάσεων και με ποιόν τρόπο μπορούν να βελτιώσουν τις υπάρχουσες μεθόδους;” Αυτό το ερώτημα πρόκειται να καλυφθεί σε αυτό το σημείο μέσα από την ανάλυση της συνεργασίας των δύο αυτών πεδίων.

2.3.1 Κοινωνικά Δίκτυα και Συστήματα Συστάσεων

Οι ρίζες της επιστήμης των δικτύων εντοπίζονται στο 1700 όπου ο Euler’s Seven Bridges of Knigsberg [8], εισήγαγε τη θεωρία των γράφων και την ανάλυση των κοινωνικών δικτύων.

Με τον όρο κοινωνικό γράφο ή δίκτυο αναφερόμαστε σε ένα δίκτυο συνδεδεμένων μελών όπου η μεταξύ τους σύνδεση αντιπροσωπεύει κάποια κοινωνική σχέση. Θα μπορούσε κάποιος να φανταστεί έναν κοινωνικό γράφο στον οποίο κάθε άτομο αποτελεί ένα κόμβο και η σύνδεση μεταξύ των κόμβων εκφράζει τη μεταξύ τους συσχέτιση. Τα κοινωνικά δίκτυα εκ φύσεως συνθέτονται από ομάδες ανθρώπων που μοιράζονται ενδιαφέροντα και δραστηριότητες. Τα παραδοσιακά συστήματα συστάσεων αγνοούσαν τις κοινωνικές σχέσεις μεταξύ των χρηστών παρά την ύπαρξη ερευνών που αποδεικνύουν την σημαντικότητα της κοινωνικής επιρροής. Τελευταία παρατηρείται όλο και συχνότερη συνεργασία των δυο αυτών πεδίων κυρίως λόγο της εμφάνισης των Social Media τα οποία έχουν τεράστια απήχηση στο κοινό.[9] Στην πραγματική ζωή όταν ζητάμε από κάποιον φίλο να μας προτείνει κάποιο ”αντικείμενο”, ουσιαστικά ζητάμε λεκτικά social recommendations. Η καρδιά ενός επιτυχημένου Συστήματος Συστάσεων είναι η δυνατότητα να

γενικεύουμε βασιζόμενοι σε γνωστές ή υπολογισμένες αξιολογήσεις αντικειμένων. Σε αυτήν ακριβός την γενίκευση βοηθούν τα Κοινωνικά Δίκτυα εκμεταλλευόμενα τη συσχέτιση των χρηστών η οποία αποκαλύπτει προτιμήσεις και συμπεριφορές. Όταν τα Συστήματα Συστάσεων χρησιμοποιούν την πληροφορία των κοινωνικών γράφων βελτιώνεται η ακρίβεια των προβλέψεων των αξιολογήσεων. [10]

2.3.2 Συστάσεις βάση Εμπιστοσύνης και Κοινωνικές Συστάσεις

Σε αυτό το σημείο θα αναφερθεί ένα άλλο συγγενές πεδίο που χρησιμοποιεί παρόμοα μέσα, για να αποφευχθούν τυχόν παρερμηνείες. Πρόσφατα, βασιζόμενες στην διαίσθηση ότι οι σχέσεις *εμπιστοσύνης* (trust) μπορούν να χρησιμοποιηθούν για την βελτίωση των παραδοσιακών συστημάτων συστάσεων, προτάθηκαν κάποιες μέθοδοι συστάσεων που βασίζονται στις δηλώσεις εμπιστοσύνης μεταξύ των χρηστών (trust-aware recommendation). Αυτές οι μέθοδοι επίσης χρησιμοποιούν τις σχετικές πληροφορίες διασύνδεσης των χρηστών για να βελτιώσουν τα παραδοσιακά συστήματα. Τέτοιου είδους συστήματα πραγματοποιούν ένα σημαντικό βήμα στην σχετική έρευνα, χωρίς όμως να πετυχαίνουν το στόχο του Social Recommendation.

Σύμφωνα με την έρευνα *Recommender Systems with Social Regularization* [10] οι μέθοδοι συστάσεων που βασίζονται στο στην εκτίμηση του βαθμού εμπιστοσύνης μεταξύ χρηστών από την εν γένει συμπεριφορά των χρηστών στο δίκτυο δεν ταυτίζονται με τις social μεθόδους αφού έχουν διαφορετική φύση. Για παράδειγμα όταν κάποιος χρήστης αρέσκεται στις αναρτήσεις κάποιου άλλου ή και όταν τελικά τον προσθέσει στην λίστα με τους ανθρώπους που παρακολουθεί ή εμπιστεύεται, δε σημαίνει απαραίτητα ότι έχει κάποια κοινωνική σχέση μαζί του. Αυτή η μονομερής διαδικασία η οποία δεν απαιτεί από τον δεύτερο χρήστη να την επιβεβαιώσει δημιουργώντας έτσι έναν κατευθυνόμενο γράφο (directed graph) μεταξύ των χρηστών. Τα πραγματικά Social Networks είναι σχεδιασμένα ώστε οι χρήστες να αλληλεπιδρούν και να συνδέονται με μέλη της πραγματικής τους κοινωνικής ζωής. Μπορούμε λοιπόν να εκτιμήσουμε ότι τα Trust-aware recommender systems δεν ταυτίζονται πάντοτε με έννοια “social” η οποία προϋποθέτει ενσωμάτωση ενός πραγματικού κοινωνικού γράφου στην εφαρμογή.

2.3.3 Πώς Θα Πετύχουμε Κοινωνικές Συστάσεις

Υπάρχουν διαφορετικές αναλύσεις για το πώς μπορεί να πραγματοποιηθεί ένα σύστημα συστάσεων βασισμένο στην κοινωνική πληροφορία των χρηστών, οι περισσότερες όμως από αυτές συμπίπτουν σε μία γενικευμένη abstract μέθοδο. Εκμεταλλευόμενοι το γεγονός πως οι άνθρωποι τείνουν να επηρεάζονται από τον κοινωνικό τους περίγυρο, οι προτιμήσεις ενός χρήστη προβλέπονται βάση των προτιμήσεων των μελών του στενού κοινωνικού τους κύκλου. Αυτή η ομάδα, στα πλαίσια της οποίας υπάρχουν σχέσεις αλληλοεπιρροής θα ονομαστεί **Κοινότητα**.

Εδώ και χρόνια η ανακάλυψη κοινοτήτων στα κοινωνικά δίκτυα αποτελεί πρόβλημα θεμελιώδους και πρακτικού ενδιαφέροντος. Αυτό συμβαίνει διότι ο άνθρωπος έχει οργανώσει το σύνολο των δραστηριοτήτων του γύρο από κοινωνικές δομές. Δυστυχώς τα υπάρχοντα εφαρμοσμένα συστήματα ομαδοποίησης υστερούν στο ότι δεν λαμβάνουν υπ όψιν την επικάλυψη των κοινοτήτων (overlap). Όλοι οι κόμβοι ομαδοποιούνται και εξάγεται γνώση από τις εσωτερικές συνδέσεις ενώ οι εκτός ομάδας συνδέσεις αγνοούνται. [11]

Συγκεκριμένα σε ένα Social Recommendation System αρχικά τα μέλη του κοινωνικού γράφου ομαδοποιούνται σε **κοινότητες** χρηστών και για κάθε χρήστη προβλέπονται οι πιθανές αξιολογήσεις, με βάση τις αξιολογήσεις που έχουν κάνει τα άλλα μέλη της κοινότητάς του. Στη συνέχεια προτείνονται στον χρήστη τα αντικείμενα με τις υψηλότερες *πιθανές* αξιολογήσεις. Η παρούσα πτυχιακή επικεντρώνεται στο στάδιο του εντοπισμού των κοινοτήτων των χρηστών παρόλα αυτά ενσωματώνει το αποτέλεσμά της σε ένα σύστημα που παράγει συστάσεις, επεκτείνοντας έτσι τη λειτουργικότητά του.

2.3.4 Τι είναι και τι αντιπροσωπεύει μια Κοινότητα Χρηστών

Η μελέτη των κοινοτικών δομών έχει μακρά ιστορία και εμφανίζεται σε πολλούς διαφορετικούς επιστημονικούς τομείς. Το τι ακριβώς αντιπροσωπεύει μια κοινότητα εξαρτάται από την εκάστοτε οπτική, αλλά συνήθως εκφράζει ένα σύνολο ανθρώπων όπου μεταξύ τους υπάρχουν ισχυρές κοινωνικές σχέσεις. Σκοπός μας είναι να εντοπίζουμε τέτοιες δομές και να εξάγουμε γνώση από αυτές. Για την επιστήμη των υπολογιστών το κοινωνικό σύνολο είναι ένα δίκτυο του οποίου οι κόμβοι έχουν το ρόλο των χρηστών και οι ακμές

που τους ενώνουν αντιπροσωπεύουν την κοινωνική τους σχέση. Οι κοινότητες είναι ουσιαστικά το αποτέλεσμα της διαίρεσης των κόμβων σε ομάδες εντός των οποίων οι συνδέσεις είναι πυκνές αλλά οι μεταξύ των ομάδων συνδέσεις αραιές. Στα περισσότερα δίκτυα, όχι μόνο στα κοινωνικά, μπορούν να εντοπιστούν κοινότητες και η εύρεση και ανάλυση τους βοηθάει στην κατανόηση των δομών των δικτύων. Ο αριθμός των κοινοτήτων που τελικά θα προκύψουν, όπως και το μέγεθος της κάθε κοινότητας στις περισσότερες των περιπτώσεων είναι άγνωστος. [12]

2.4 Αλγόριθμοι για την παραγωγή Κοινοτήτων Χρηστών

Στο πρόγραμμά μας η ένταξη των χρηστών σε κοινωνικές ομάδες γίνεται με χρήση αλγορίθμων **clustering** και **partitioning** οι οποίοι εντοπίζουν κοινότητες χρηστών βασιζόμενοι στις κοινωνικές τους συνδέσεις ο κάθε ένας με διαφορετικό τρόπο. Clustering είναι η διαδικασία *ομαδοποίησης* των αντικειμένων (χρήστες) με τέτοιο τρόπο ώστε τα μέλη μιας ομάδας να εμφανίζουν κάποιου είδους ομοιότητα ενώ τα αντικείμενα διαφορετικών ομάδων ανομοιότητα. Partitioning είναι η διαδικασία κατάτμησης του γράφου σε μικρότερες δομές που έχουν συγκεκριμένα χαρακτηριστικά.

Διαισθητικά στην περίπτωσή μας, ένα cluster είναι μία συλλογή αντικειμένων με πυκνή κοινωνική συσχέτιση στο εσωτερικό τους και αραιή στο εξωτερικό τους. Υπάρχουν πολλά κριτήρια που καθορίζουν μια διαδικασία clustering ως καλά πραγματοποιημένη όπου το κάθε ένα συνοδεύεται από πλήθος αλγορίθμων. [11]

Μια από τις πιο χρήσιμες προσεγγίσεις, δανεισμένη από το πεδίο του social network analysis, εφαρμόζει ένα σύνολο τεχνικών γνωστών ως: **Ιεραρχικό clustering**. Αυτές οι τεχνικές στοχεύουν στην ανακάλυψη φυσικών διαιρέσεων του κοινωνικού γράφου, βασιζόμενες σε διάφορες μετρικές ομοιότητας ή βάρους των συνδέσεων μεταξύ των κόμβων. Υπάγονται σε δύο γενικές κατηγορίες: *Συγκεντρωτικές* και *Διαιρετικές*, ανάλογα με το αν πραγματοποιούν πρόσθεση ή αφαίρεση κόμβων στο δίκτυο.

Συγκεντρωτικές - Agglomerative μέθοδοι

Αρχικά υπολογίζονται οι ομοιότητες μεταξύ των ζευγών κόμβων με κάποια μέθοδο και στην συνέχεια τα ζεύγη προστίθενται σε ένα αρχικά άδειο δίκτυο αρχίζοντας με τα ζεύγη που έχουν τη μεγαλύτερη ομοιότητα. Η διαδικασία μπορεί να τερματιστεί ανά πάσα στιγμή και οι κόμβοι που έχουν μπει στο νέο δίκτυο αποτελούν τις κοινότητες. Εναλλακτικά η όλη διαδικασία μπορεί να παρουσιαστεί με ένα δέντρο (ή δενδρόγραμμα), όπου οι οριζόντιες τομές του δέντρου αντιπροσωπεύουν τις κοινότητες. Γενικά χρησιμοποιείται μεγάλη ποικιλία μετρικών ομοιότητας.
Κάποιες προβληματικές τους είναι ότι συχνά αποτυγχάνουν να βρουν τις σωστές κοινότητες σε δίκτυα όπου η κοινοτική δομή είναι ήδη γνωστή, πράγμα που τις κάνει να χάνουν την εμπιστοσύνη τους. Ένα άλλο πρόβλημα είναι πως έχουν την τάση να βρίσκουν μόνο τους πυρήνες των κοινοτήτων και αδυνατούν να εντοπίσουν τους περιφερειακούς κόμβους. Οι κόμβοι που αποτελούν τον πυρήνα έχουν μεγάλη ομοιότητα και ως εκ τούτου συνδέονται από την αρχή της agglomerative διαδικασίας, αλλά έτσι οι περιφερειακοί κόμβοι που δεν έχουν ισχυρή ομοιότητα παραμελούνται.

Διαιρετικές - Divisive μέθοδοι

Δεν έχουν μελετηθεί πολύ στην σχετική βιβλιογραφία, αλλά είναι πολλά υποσχόμενες. Μία διαιρετική μέθοδος ξεκινά ψάχνοντας τα ζεύγη κόμβων με τις *λιγότερες* ομοιότητες και στη συνέχεια αφαιρεί τη μεταξύ τους σύνδεση. Επαναλαμβάνοντας την διαδικασία το δίκτυο καταλήγει να διαιρείται σε όλο και πιο μικρές ομάδες και τερματίζοντας την διαδικασία στο επιθυμητό σημείο αποκτούνται οι κοινότητες. Μπορεί να παρουσιαστεί σε δενδρόγραμμα το οποίο απεικονίζει τις επιτυχείς διαιρέσεις του δικτύου σε όλο και μικρότερες ομάδες. [12]

2.5 Ανάλυση των Αλγορίθμων

Επιλέχθηκε να χρησιμοποιηθεί ένας αριθμός αλγορίθμων οι οποίοι μπορούν να εφαρμοστούν σε κοινωνικούς γράφους και έχουν νόημα για την συγκεκριμένη περίπτωση.

WeakComponent
: εντοπίζει ”κλίκες” στενά συνδεδεμένων χρηστών

EdgeBetweenness
ομαδοποιεί τους κόμβους του γράφου με βάση τη θέση τους στα ”μονοπάτια” τους ενώνουν. Υπολογίζει την μετρική ανάλυσης κοινωνικών δικτύων Betweenness

BronKerbosch
εντοπίζει τις μέγιστες ”κλίκες” χρηστών

Metis
Σύνολο ισχυρών εργαλείων κατάτμησης γράφων που αναπτύσσεται από μέλη της ακαδημαϊκής κοινότητας

Οι περισσότεροι υπάρχοντες αλγόριθμοι τέτοιου είδους πραγματοποιούν πλήρως διαζευκτικό clustering, πράγμα που σημαίνει πως ο κάθε κόμβος (χρήστης) ανήκει σε ένα μόνο cluster. Αυτή η προσέγγιση δεν είναι πλήρως αποδοτική αφού δεν είναι ρεαλιστική καθώς στην πραγματική ζωή κάποιος μπορεί να ανήκει (και συνεπώς να επηρεάζεται) σε πολλές κοινωνικές ομάδες ή και σε καμία. Μέχρι στιγμής έχει πραγματοποιηθεί σχετικά μικρή έρευνα στο πεδίο της επικαλυπτόμενης ομαδοποίησης (overlapping clustering). [13]

Οι υλοποιήσεις των αλγορίθμων που ενσωματώθηκαν προέρχονται από java βιβλιοθήκες ανοιχτού κώδικα οι οποίες αποτελούν είτε έρευνα είτε εμπορικές εφαρμογές. Ακολουθεί η ανασκόπηση και περαιτέρω περιγραφή αυτών.

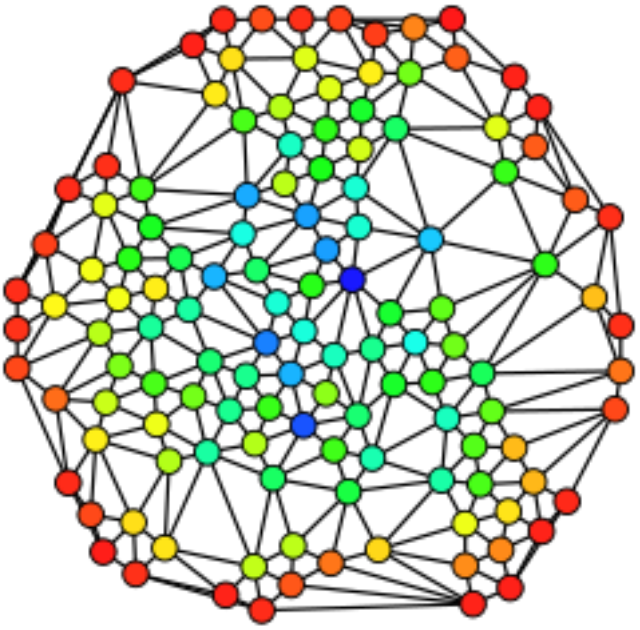
2.5.1 WeakComponent

Εντοπίζει τα weakly connected components σε ένα κοινωνικό γράφο. Ένα weakly connected component είναι ένα σύνολο κόμβων (χρηστών) όπου ο κάθε κόμβος έχει μονοπάτι για κάθε άλλο κόμβο στο σύνολο. Αυτή η μορφή κοινότητας πολλές φορές αναφέρεται και ως **κλίκα**. Ο συγγραφέας του Scott White [14], αναφέρει πως Weak Component είναι ο μέγιστος υπογράφος (subgraph) στον οποίον όλα τα ζεύγη χρηστών είναι προσβάσιμα μεταξύ τους.

Η πολυπλοκότητα εκτέλεσης του αλγορίθμου είναι: O(|V| + |E|) όπου |V| είναι ο αριθμός των χρηστών και |E| είναι ο αριθμός των ακμών φιλίας.

2.5.2 EdgeBetweenness

Ο αλγόριθμος αυτός βασίζεται στη έννοια του *'"Ενδιάμεσου" (Betweenness)* η οποία εισάχθηκε σαν εργαλείο ανάλυσης κοινωνικών δικτύων από τον κοινωνιολόγο Linton Freeman [15] σύμφωνα με τον οποίον σε ένα κοινωνικό δίκτυο ένας κόμβος είναι σημαντικός και έχει υψηλό betweenness centrality όταν εμπεριέχεται σε πολλά συντομότερα μονοπάτια (sortest paths) μεταξύ των κόμβων του δικτύου.



Σχήμα 2.1: Η απόχρωση (από red=0 ως blue=max)
δείχνει το betweenness του κάθε κόμβου στο γράφο. [16]

Στη συνέχεια οι M.Girvan και J.Newman [17] επεκτείνουν την μεθοδολογία εισάγοντας το edge betweenness, που και πάλι μέτρα centrality αλλά για τις ακμές του γράφου. Παράλληλα εισάγουν τον αλγόριθμο Girvan–Newman ο οποίος στοχεύει στην εξαγωγή κοινοτήτων από έναν κοινωνικό γράφο βασιζόμενος στο betweenness των ακμών. Παραλλαγή αυτού είναι και ο αλγόριθμος που ενσωματώσαμε στον PServer.

Όπως αναφέρουν οι συγγραφείς του, Scott White και Tom Nelson [18], ο EdgeBetweenness είναι ένας αλγόριθμος clustering για την εύρεση κοινοτήτων χρηστών σε γράφους που βασίζεται στο betweenness των ακμών. Ως betweenness ορίζεται ο βαθμός στον οποίο μία ακμή βρίσκεται στα shortest paths μεταξύ όλων των ζευγών κόμβων. Ο αλγόριθμος αυτός λειτουργεί επαναληπτικά ακολουθώντας μια διαδικασία δύο βημάτων:
- υπολογισμός του edge betweenness όλων των ακμών του γράφου

- διαγραφή των ακμών με το υψηλότερο betweenness.

Ο αλγόριθμος αυτός διαφοροποιείται από τον Girvan–Newman όσον αφορά τον αριθμό των ακμών που διαγράφονται σε κάθε βήμα, ο οποίος πλέον αποτελεί παράμετρο.

Η πολυπλοκότητα εκτέλεσης του αλγορίθμου είναι: $O(kmn)$ όπου *k* ο αριθμός των ακμών που θα διαγραφούν, *m* ο αριθμός των ακμών, και *n* ο αριθμός των χρηστών (κόμβων). Για αραιούς γράφους ο χρόνος εκτέλεσης πλησιάζει στο $O(kn^2)$ και για πυκνούς γράφους η πολυπλοκότητα είναι ακόμα μικρότερη.
Τελικά ο αλγόριθμος εντοπίζει τις ομάδες χρηστών (communities) που έχουν την υψηλότερη κοινωνική δομή. Όσες περισσότερες ακμές διαγραφούν τόσο μικρότερα και συνεκτικά τα clusters.

2.5.3 BronKerbosch

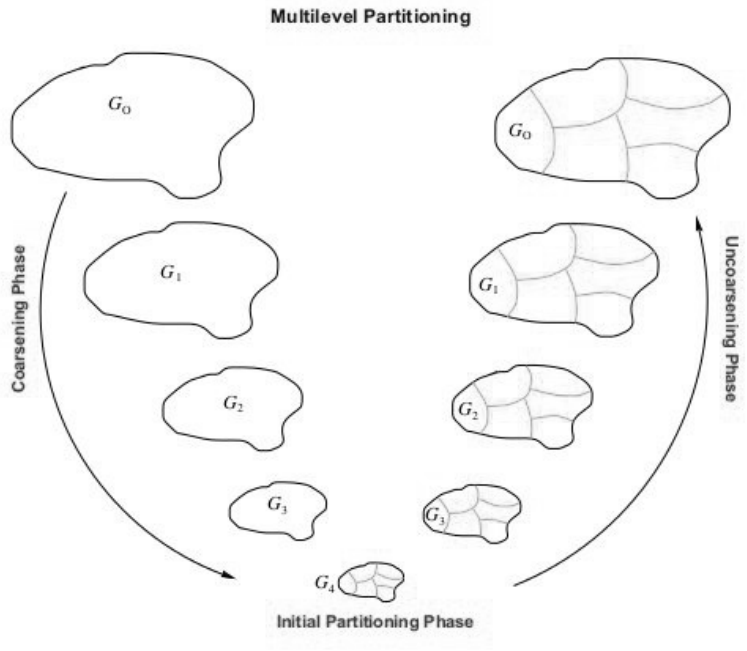
Ο Bron-Kerbosch είναι ένας clique detection αλγόριθμος που αναπτύχθηκε από τους Bron και Kerbosch το 1973 [19]. Στη συνέχεια οι Samudrala και Moult [20] τον προσέγγισαν σαν μια αναδρομική *backtracking* διαδικασία χρησιμοποιώντας επίσης μία *branch and bound* τεχνική ώστε να εξαλειφθούν οι αναζητήσεις που δεν οδηγούν σε κλίκα. Η αναδρομική διαδικασία είναι αυτοαναφορική με την έννοια ότι η εύρεση μίας κλίκας μεγέθους n επιτυγχάνεται με το να βρίσκεται μια κλίκα μεγέθους $n-1$ και στη συνέχεια να εντοπίζεται κάποιος κόμβος που να είναι συνδεδεμένος με όλους τους κόμβους μέλη της κλίκας. Η branch and bound τεχνική κάνει δυνατή τη χρήση κανόνων για εκ των προτέρων καθορισμό των περιπτώσεων στις οποίες οι πιθανοί συνδυασμοί κόμβων και ακμών δεν θα οδηγήσουν ποτέ σε κλίκα.

Σε αυτήν την προσέγγιση έχει βασίσει την υλοποίησή του και ο συγγραφέας της έκδοσης που ενσωματώσαμε στο πρόγραμμά μας Ewgenij Proschak [21], ο οποίος αναφέρει για τον αλγόριθμο πως σκοπός του είναι να βρει όλες τις μέγιστες κλίκες (maximal cliques) του γράφου. Μια κλίκα είναι μέγιστη αν το μέγεθός της είναι αδύνατο να αυξηθεί με την πρόσθεση κάποιου ακόμα κόμβου του γράφου. Επισημαίνεται, πως μια μέγιστη κλίκα δεν είναι απαραίτητα και η μεγαλύτερη κλίκα του γράφου.

2.5.4 Metis

Η βιβλιοθήκη METIS [22] προσφέρει μία συλλογή προγραμμάτων γραμμένων σε C/C++ και Fortran για την κατάτμηση γράφων (graph partitioning) και δικτύων πεπερασμένων στοιχείων.

Συγκεκριμένα για την παραγωγή communities στον socialPServer χρησιμοποιούμε το εργαλείο GpMetis του οποίου η λειτουργία είναι η κατάτμηση γράφων σε k ίσα μέρη, όπου το πλήθος k καθορίζεται από τον χρήστη. Οι αλγόριθμοι που είναι υλοποιημένοι χρησιμοποιούν είτε multilevel recursive bisection είτε multilevel k-way partitioning paradigms (σχήμα 2.2). Και οι δύο αυτοί μέθοδοι είναι σε θέση να παράγουν υψηλής ποιότητας partitions, παρόλα αυτά η multilevel k-way partitioning προτιμάται γιατί παρέχει κάποιες περαιτέρω δυνατότητες όπως: (Ελαχιστοποίηση το παραγόμενου υπογράφου, συνεχόμενες κατατμήσεις, και άλλα.) [23]



Σχήμα 2.2: Οι τρεις φάσεις του multilevel k-way graph partitioning. Κατά την διάρκεια της συμπίεσης (coarsening), το μέγεθος του γράφου μειώνεται επιτυχώς. Κατά την διάρκεια του initial partitioning, υπολογίζεται ένα k-way partitioning. Κατά την διάρκεια της multilevel refinement (ή uncoarsening) φάσης, οι κατατμήσεις αποσυμπιέζονται διαδοχικά και προβάλλονται σε όλο και μεγαλύτερους γράφους. G0 είναι ο input γράφος, ο οποίος είναι ο λιγότερο συμπιεσμένος. Gi+1 είναι ο αμέσως πιο συμπιεσμένος γράφος και ο G4 είναι ο πιο συμπιεσμένος γράφος που θα υπάρξει. [23]

Η ενσωμάτωση του Metis στον java κώδικα μας πραγματοποιείτε με τη χρήση της κλάσης Gpmetis [24] της βιβλιοθήκης χειρισμού γράφων Grph, η οποία αναλαμβάνει τη μεταγλώττιση (compile) του C κώδικα σε πραγματικό χρόνο, την εκτέλεση των διαδικασιών και τον χειρισμό του I/O.

Κεφάλαιο 3

Σχεδιασμός

3.1 socialPServer

Όπως προαναφέρθηκε, η προσέγγιση που ακολουθάτε στην παρούσα εργασίας αφορά την επέκταση ενός συστήματος συστάσεων ώστε να μπορεί να εξάγει συμπεράσματα για τους χρήστες του μελετώντας τον κοινωνικό τους γράφο. Με αυτό τον τρόπο θα βελτιωθεί η ακρίβεια των συστάσεων με παραδοχή πως οι χρήστες τείνουν να έχουν κοινά ενδιαφέροντα και προτιμήσεις με τους ανθρώπους τους οποίους έχουν επιλέξει να είναι κοινωνικά συνδεδεμένοι. Αυτό πρακτικά σημαίνει πως έχοντας κάποιας μορφής κοινωνικό γράφο μπορούν να ομαδοποιηθούν τα μέλη του (χρήστες) σύμφωνα με τις ακμές φιλίας ή εμπιστοσύνης που τους ενώνουν. Εντοπίζονται δηλαδή ομάδες (communities) *στενά συνδεδεμένων χρηστών* οι οποίες αποτελούν πηγή εξόρυξης διαφόρων τύπων δεδομένων. Συγκεκριμένα, πρόκειται να αξιοποιηθεί η ύπαρξη κάποιου χρήστη σε κάποια κοινωνική ομάδα για να σκιαγραφηθούν οι προτιμήσεις του.

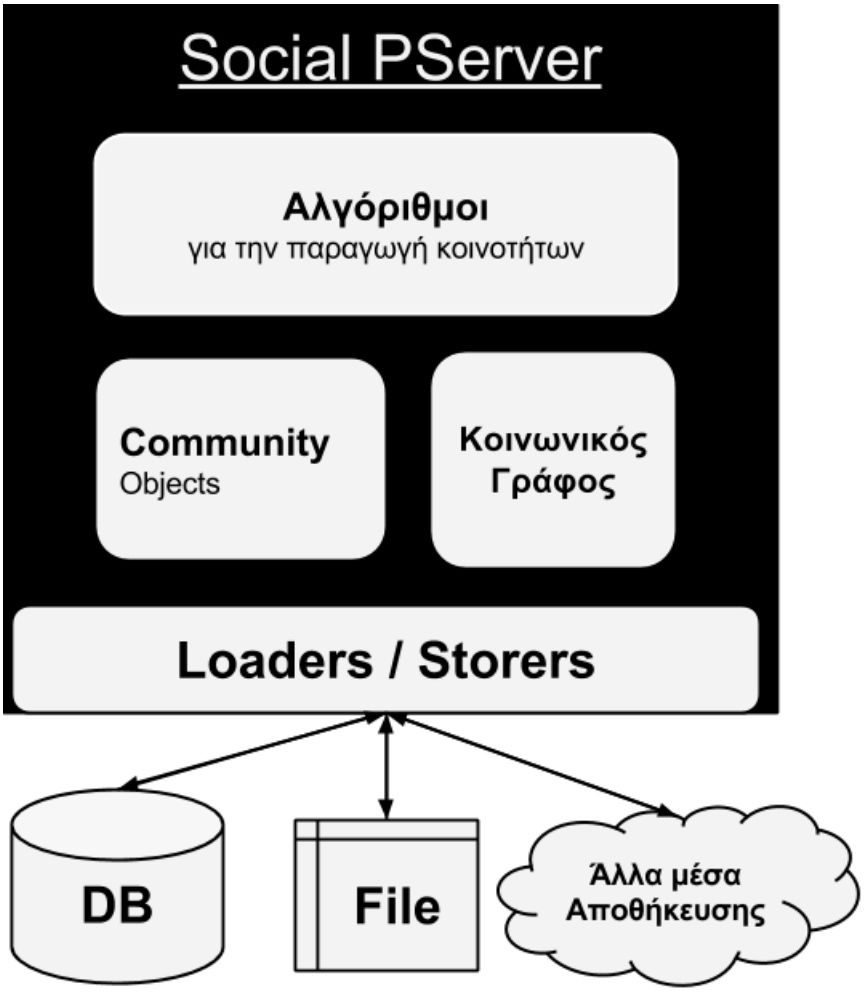
Το πρόγραμμα που υλοποιήθηκε για την επίτευξη των παραπάνω *(social PServer)* έχει τους παρακάτω **στόχους**:

Να αποτελέσει μια πλατφόρμα στην οποία μπορούν να ενσωματωθούν αλγόριθμοι παραγωγής κοινοτήτων χρηστών δοσμένου ενός κοινωνικού γράφου.
Επομένως πρέπει να περιλαμβάνονται όλες οι απαραίτητοι μέθοδοι για τον χειρισμό τον δεδομένων και την ολοκλήρωση της διαδικασίας διατηρώντας μια εσωτερική αυτονομία των λειτουργιών. Αυτό επιτυγχάνεται διατηρώντας σαφή είσοδο και έξοδο της κάθε διαδικασίας και

δημιουργώντας τα κατάλληλα *interfaces* που θα λειτουργήσουν σαν πρότυπα-οδηγοί για την προσθήκη νέων αλγορίθμων.

Να εγκατασταθεί ένας αριθμός αλγορίθμων στο πρόγραμμα.
Ώστε να δοκιμαστεί η διαδικασία αλλά κυρίως για να αποκτήσει λειτουργικότητα. Η ενσωμάτωση αλγορίθμων στο πρόγραμμα το καθιστά πλέον όντος μια μηχανή εντοπισμού κοινοτήτων η οποία είναι εργαλείο για ανάλυση κοινωνικών δικτύων και μπορεί να εφαρμοστεί σε πληθώρα εφαρμογών από την κοινωνιολογία, ως την στοχευμένη διαφήμιση.

Να ενσωματωθεί στο υπάρχον Recomendation System PServer
Να χρησιμοποιηθεί δηλαδή το πρόγραμμά μας για να επεκτείνει την λειτουργία του PServer επιτρέποντας του να παρέχει προσωποποίηση αναλύοντας τον κοινωνικό γράφο των χρηστών του.



Σχήμα 3.1: Η εσωτερική οργάνωση του socialPServer

3.2 Κοινωνική Πληροφορία

Αρχικό βήμα για την επίλυση του προβλήματος είναι να μπορούν να διαβαστούν και να επεξεργαστούν οι κοινωνικές πληροφορίες. Το ποιες πληροφορίες πρέπει να συλλεχθούν στην την κάθε περίπτωση εξαρτάται από τη φύση αλλά και τους στόχος της υπηρεσίας. Η συλλογή αυτών δεν αφορά τον socialPServer αλλά πραγματοποιείται από την εκάστοτε υπηρεσία. Κοινωνική συσχέτιση μπορεί να σημαίνει ξεκάθαρη δήλωση φιλίας από τον χρήστη άλλα μπορεί και να εντοπίζεται μέσα από την αλληλεπίδραση του χρήστη με το σύστημα. Δηλαδή η κοινωνική πληροφορία μπορεί να εκφράζει διαφορετικά πράγματα, αλλά αυτό που σε κάθε περίπτωση έχει ενδιαφέρον είναι η αξιοποίηση της κοινωνικής διασύνδεσης.

Η σχέση μεταξύ δυο χρηστών μπορεί να έχει διάφορες μορφές και χαρακτηριστικά. Μπορεί να είναι να έχει κατεύθυνση (directed) που σημαίνει πως κάποιος χρήστης εμπιστεύεται κάποιον άλλον όμως μπορεί να μην συμβαίνει το αντίθετο, μπορεί να έχει κάποιο βάρος (weighted) το οποίο υποδηλώνει τον βαθμό φιλίας των δύο ή μπορεί και να είναι undirected και unweighted να δηλώνει δηλαδή απλά την συσχέτιση δύο χρηστών. Στην τελευταία αυτή κατηγορία έγκειται και το πρόγραμμά που υλοποιήσαμε (socialPServer) ο οποίος αποθηκεύει την εκάστοτε φιλία στην Βάση Δεδομένων καταχωρώντας μία εγγραφή τα userIDs των δύο εμπλεκόμενων χρηστών.

Κοινωνικός Γράφος

Σε αυτό το σημείο, έχοντας λοιπόν την κοινωνική πληροφορία, αρχίζει η λειτουργία του socialPServer. Το πρώτο πράγμα που πρέπει να γίνει είναι η ανάκτηση των δεδομένων φιλίας και η δημιουργία μιας δομής δεδομένων η οποία να αντιπροσωπεύει έναν εικονικό κοινωνικό γράφο. Για λόγους τόσο ευχρηστίας αλλά και αποδοτικότητας, για τον χειρισμό των γράφων, χρησιμοποιούνται open-source java βιβλιοθήκες στοχευμένες σε αυτό. Στη συνέχεια λοιπόν ανάλογα με τον αλγόριθμο που θα χρησιμοποιηθεί για την παραγωγή κοινοτήτων, χρησιμοποιείται και η κατάλληλη κοινωνική δομή - γράφος.

3.3 Κοινότητες χρηστών

Με την εφαρμογή των αλγορίθμων πρόκειται να εντοπιστούν σύνολα χρηστών τα οποία μπορούν να χαρακτηριστούν κοινότητες. Για να είναι αυτή η γνώση αξιοποιήσιμη, σε κάθε μία από αυτές δίνεται ένα ID και στην συνέχεια για κάθε μέλος ομάδας, αποθηκεύεται στην Βάση μια εγγραφή που φέρει το UserID και το CommunityID. Με αυτόν το τρόπο μπορεί ο διαχειριστής της εκάστοτε υπηρεσίας με μία ερώτηση στη βάση (query) να γνωρίζει ποιοι χρήστες αποτελούν ομάδα, ποιος είναι ο κοινωνικός κύκλος του κάθε χρήστη και λοιπές πληροφορίες.

3.4 Προφίλ Χρήστη (userProfile)

Για να μπορούμε τελικός να παραχθούν συστάσεις, πρέπει εκτός από την κοινωνική πληροφορία να είναι γνωστές και οι προτιμήσεις των χρηστών, που είναι οι υπάρχουσες αξιολογήσεις αντικειμένων. Επομένως πρέπει να υπάρχει ένα *προφίλ χρήστη* το οποίο θα περιέχει τις κάθε φορά απαραίτητες πληροφορίες που μας επιτρέπουν να σκιαγραφήσουμε τα ενδιαφέροντα και την συμπεριφορά του. Στην περίπτωσή μας αυτή η διαδικασία πραγματοποιείται είδη στον PServer, αφού πρόκειται για μια υπάρχουσα, λειτουργική και εφαρμοσμένη μηχανή προσωποποίησης. Συγκεκριμένα το *user profile* περιλαμβάνει πληροφορίες για τις προτιμήσεις των χρηστών αλλά και για τα δημογραφικά τους χαρακτηριστικά. Στα πλαίσια αυτής της εργασίας χρησιμοποιούμε τις πληροφορίες των προφίλ για να αξιολογούμε την διαδικασία παραγωγής κοινοτήτων χρησιμοποιώντας υπάρχοντα δεδομένα (εσωτερική αξιολόγηση). Οι προτιμήσεις των χρηστών θα ξαναχρησιμοποιηθούν όταν θα γίνουν τελικά οι συστάσεις αλλά αυτό όπως φαίνεται και από την παραπάνω περιγραφή του συστήματος, είναι λειτουργία του εκάστοτε recommendation engine που ανάλογα με την περίπτωση θα υπολογίσει τις πιθανές αξιολογήσεις.

3.5 PServer

Σε αυτό το σημείο θα δοθεί περαιτέρω εξήγηση για την υπάρχουσα πλατφόρμα την οποία καλείτε να επεκτείνει το δικό μας πρόγραμμα. Ο PServer[25] είναι ένας γενικής χρήσης *personalization Server* οποίος είχε αναπτυχθεί

από το ερευνητικό κέντρο **Δημόκριτος** και στην συνέχεια από τον οργανισμό **SciFY**. Επόμενος πρόκειται για να λειτουργικό πρόγραμμα που μπορεί να εγκατασταθεί σε διαφορετικών ειδών εφαρμογές για να προσφέρει προσωποποιημένη υπηρεσία. Με τον όρο προσωποποίηση εννοούμε διαφοροποίηση του περιεχομένου μιας υπηρεσίας ανάλογα με τις προτιμήσεις και την αισθητική του κάθε χρήστη. [26]
Προσωποποίηση επιτυγχάνεται με διαφορετικούς τρόπους όπως εμφάνιση διαφορετικού GraphicalUserInterface σε κάθε χρήστη ώστε να είναι προσιτό σε αυτόν, προβολή διαφημίσεων που αφορούν μόνο αντικείμενα για τα οποία χρήστης έχει δείξει ενδιαφέρον στο παρελθόν και άλλους. Ο PServer περιλαμβάνει τις λειτουργίες που χρειάζονται για να υπάρξει ένα σύστημα συστάσεων. Αποτελεί δηλαδή την μηχανή η οποία θα αναλάβει να να οργανώσει τα προφίλ των χρηστών και να τους κατηγοριοποιήσει ώστε να μπορέσουν να εξαχθούν συμπεράσματα για τις προτιμήσεις τους.

Η λειτουργία του περιλαμβάνει τα εξής λογικά επίπεδα.

Personal user models
: Είναι ουσιαστικά οι πληροφορίες που αποθηκεύονται για κάθε χρήστη ώστε να μπορεί να δημιουργηθεί ένα user profile. Μπορεί να είναι δημογραφικά χαρακτηριστικά (*Attributes* - πχ. ηλικία, φύλο) ή πληροφορίες περιεχομένου (*Features*) που αναφέρονται στην οντολογία της κάθε εφαρμογής (πχ αξιολογήσεις αντικειμένων).

Stereotypes
: Είναι ομάδες χρηστών με κοινά δημογραφικά χαρακτηριστικά (Attributes), οι οποίες παράγονται από τον PServer.

User Communities
: Ομάδες χρηστών που προκύπτουν από την αλληλεπίδρασή τους με το σύστημα. Προς το παρόν οι χρήστες ομαδοποιούνται από τον PServer με βάσει τα Feature τους. Σε αυτό το λογικό επίπεδο ανήκει και το πρόγραμμά μας, το οποίο πρόκειται να εντοπίζει τέτοιου είδους ομάδες μελετώντας την κοινωνική συσχέτιση των χρηστών.

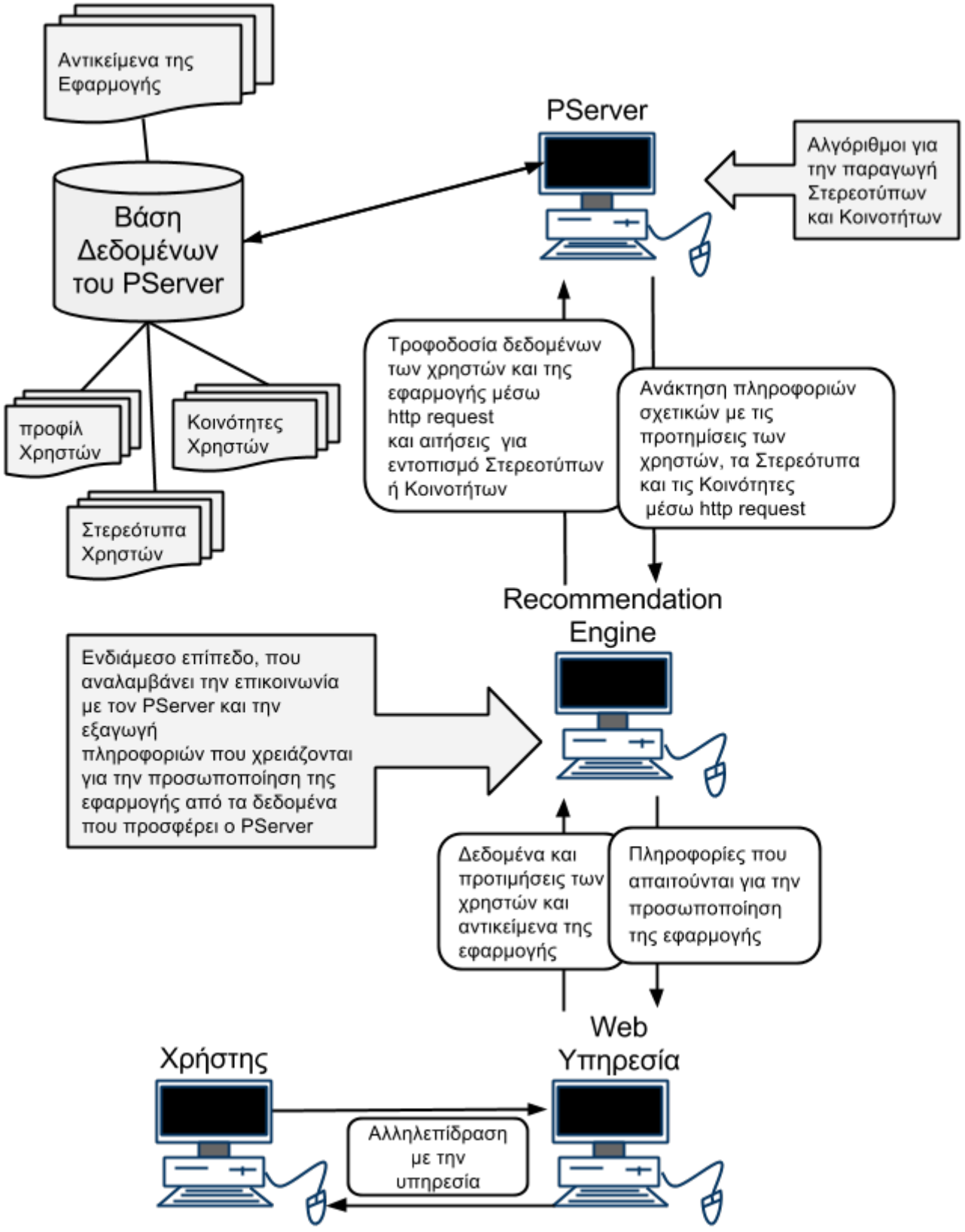
[26]

3.6 Αρχιτεκτονική του PServer

Ο PServer έχει σχεδιαστεί ώστε να είναι ανεξάρτητος από την υπηρεσία και την πλατφόρμα, με την έννοια ότι μπορεί να λειτουργήσει σε διαφορετικά Λειτουργικά Συστήματα και μπορεί να προσφέρει υπηρεσίες προσωποποίησης σε κάθε εφαρμογή ανεξαρτήτως περιεχομένου. Για την επίτευξη των παραπάνω, η υλοποίησή του έχει γίνει σε java χωρίς καθόλου χρήση API του λειτουργικού συστήματος. Το μόνο που απαιτείται είναι το μηχάνημα να διαθέτει ένα port με Java virtual machine version 1.5+. Επίσης για να λειτουργήσει χρειάζεται μία σχεσιακή βάση δεδομένων, συγκεκριμένα χρησιμοποιεί MySQL 5+, η οποία είναι ανοιχτού κώδικα και λειτουργεί σε όλα τα διαδεδομένα Λειτουργικά Συστήματα. Τέλος για να διατηρεί ανεξαρτησία από την πλατφόρμα είναι σχεδιασμένος ώστε να επικοινωνεί χρησιμοποιώντας HTTP πρωτόκολλο. Δέχεται Http requests από τις εφαρμογές που εξυπηρετεί, τα οποία περιλαμβάνουν εντολές και παραμέτρους και δίνει απάντηση σε μορφή XML ή JSON. [26]

Ο PServer μπορεί να εξυπηρετεί ταυτόχρονος πολλές WEB υπηρεσίες τις οποίες ξεχωρίζει με ένα *ClientID*. Κάθε εγγραφή που αποθηκεύεται στην βάση έχει και μια στήλη με αυτό το χαρακτηριστικό ώστε να ξεχωρίζεται το περιεχόμενο των διαφορετικών πελατών του. Σε κάθε HTTP request, εμπεριέχονται επίσης το ClientID ένας κωδικός για την αυθεντικοποίηση της υπηρεσίας.

Κάθε υπηρεσία κάνει διαφορετικές ενέργειες ώστε να προσωποποιήσει το περιεχόμενό της. Αυτό που προσφέρει ο PServer είναι οι απαραίτητες πληροφορίες που απαιτούνται για να υπάρξει ένα σύστημα προσωποποίησης. Για να μπορέσει μια υπάρχουσα υπηρεσία να εκμεταλλευτεί την λειτουργία του, πρέπει να υλοποιηθεί ένα ενδιάμεσο επίπεδο επικοινωνίας μεταξύ του PServer και της υπηρεσίας που ονομάζεται **Recommendation Engine**, του οποίου ο σχεδιασμός εξαρτάται κάθε φορά από τους στόχους της εφαρμογής. Αυτό το επίπεδο αναλαμβάνει να τον τροφοδοτεί με δεδομένα, να δώσει τις σχετικές εντολές που χρειάζονται για την παραγωγή Στερεοτύπων και Κοινοτήτων και να ανακτήσει πληροφορίες από αυτόν. Επίσης εδώ θα προεπεξεργαστούν οι πληροφορίες που επιστρέφει ο PServer, ώστε να έχουν νόημα για την εκάστοτε υπηρεσία.



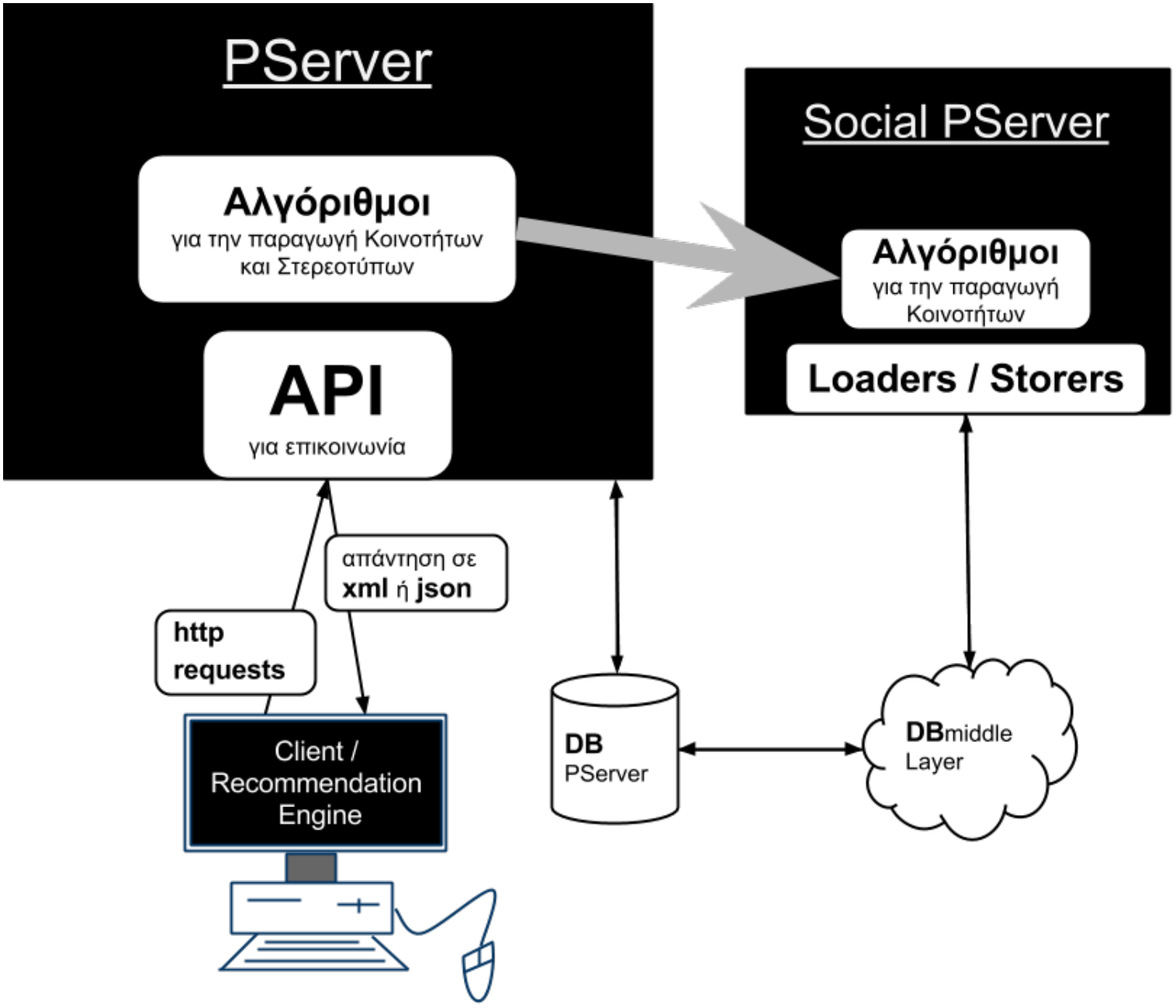
Σχήμα 3.2: Η αρχιτεκτονική του PServer

3.7 ενσωμάτωση στον PServer

Σημαντικός παράγοντας της αρχιτεκτονικής του προγράμματός μας είναι η αυτονομία και η μεταφερσιμότητα, ώστε να μπορεί να συμπεριληφθεί σε οποιαδήποτε άλλη java εφαρμογή σαν βιβλιοθήκη. Έτσι και στην περίπτωση του PServer ενσωματώθηκε σαν βιβλιοθήκη και για να γίνει χρήση των λειτουργιών του καλείται με τις ανάλογες παραμέτρους που αφορούν την επιλογή του αλγορίθμου και τις παραμέτρους αυτού.

Η λειτουργία του socialPServer απαιτεί ένα μέσο αποθήκευσης Δεδομένων στο οποίο να εμπεριέχονται τα στοιχεία των χρηστών και της εφαρμογής, επομένως κατά την εγκατάστασή του σε κάποιο μηχάνημα ο προγραμματιστής δημιουργεί την σχετική Βάση Δεδομένων ή κάποιου τύπου αρχείο με τις απαραίτητες πληροφορίες. Εφόσον σκοπός είναι να επεκταθούν οι υπάρχουσες λειτουργίες του PServer καλύτερο θα ήταν να χρησιμοποιεί κοινή Βάση Δεδομένων με αυτόν. Για να μπορούν λοιπόν να συνεργάζονται σε επίπεδο δεδομένων, στο πρόγραμμά μας χρησιμοποιούνται οι υπάρχουσες μέθοδοι που είναι υλοποιημένες στον pServer για την αλληλεπίδραση με την Βάση. Επομένως δημιουργήσαμε ένα ενδιάμεσο επίπεδο που μεταφράζει την επικοινωνία μας με την Βάση στις μεθόδους επικοινωνίας του PServer. Πρακτικά λοιπόν όταν ο pServer καλεί το πρόγραμμά μας δίνει ως παράμετρο και ένα αντικείμενο της κλάσης του *DBAccess* το οποίο περιλαμβάνει μεθόδους για τον χειρισμό των πινάκων με τους οποίους λειτουργεί. Αυτό το αντικείμενο δίνουμε κατά την αρχικοποίηση στην δικιά μας ενδιάμεση κλάση η οποία θα αναλάβει να μεταφράζει όλες τις επικοινωνίες του προγράμματός μας σε επικοινωνίες του pServer.

Τέλος, στην πλευρά του PServer, προστέθηκαν κάποιες λειτουργίες στο API του χρησμού του, ώστε να μπορεί η κάθε υπηρεσία που τον χρησιμοποιεί να τον τροφοδοτήσει με την απαραίτητη πληροφορία φιλίας αλλά και να χρησιμοποιεί τους Αλγορίθμους του.
Συγκεκριμένα ο PServer ενημερώθηκε καταλλήλως για να μπορεί να δεχτεί αιτήσεις που δηλώνουν την φιλία χρηστών και αιτήσεις για παραγωγή κοινοτήτων χρηστών με βάση την κοινωνική πληροφορία.



Σχήμα 3.3: ενσωμάτωση του socialPServer στον PServer
Το βέλος που ενώνει τους δυο μηχανισμούς Αλγορίθμων συμβολίζει την περίπτωση που ο Client, μέσω του API, ζητά τον εντοπισμό κοινοτήτων με βάση τους κοινωνικούς δεσμούς και επομένως
ο PServer καλεί τον socialPServer

Κεφάλαιο 4

Υλοποίηση

Σε αυτό το κεφάλαιο θα γίνει μια ανασκόπηση της υλοποίησης του socialpserver , θα περιγραφούν ο java κώδικας, οι διαφορετικές κλάσεις και οι μέθοδοι που αναπτύχθηκαν. Οι κλάσεις που χρησιμοποιούνται είναι χωρισμένες στα εξής πακέτα.

socialpserver 4.1
: Το οποίο περιλαμβάνει τις βασικότερες για την λειτουργία, κλάσεις του socialPServer.

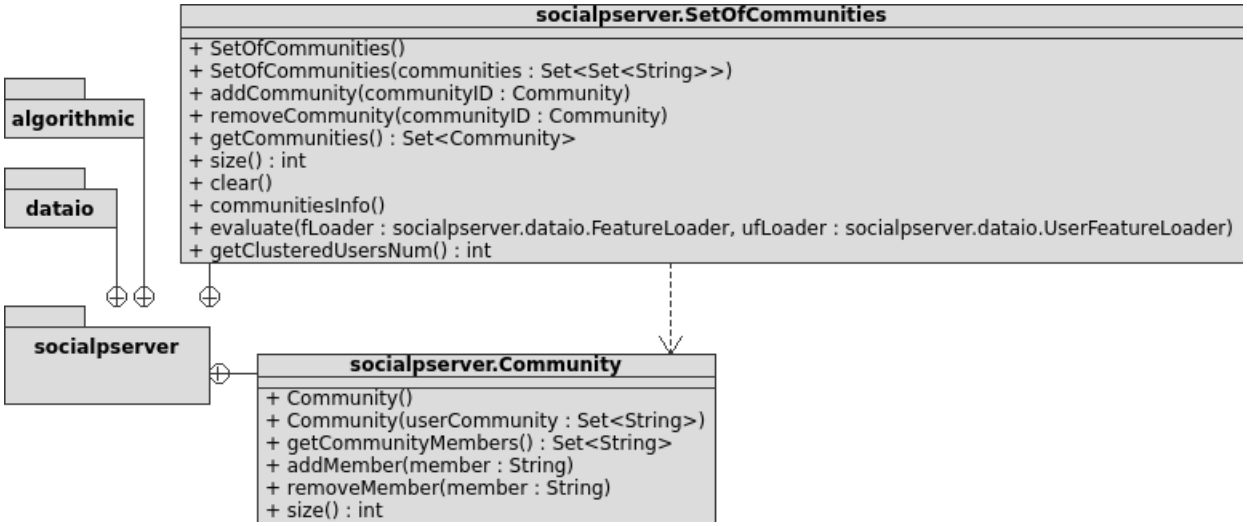
socialpserver.algorithmic 4.2
: Στο οποίο βρίσκονται κλάσεις σχετικές με τους αλγορίθμους που χρησιμοποιούνται για την παραγωγή communities.

socialpserver.dataio 4.3
: Που περιλαμβάνει όλες τις κλάσεις που σχετίζονται με την είσοδο και έξοδο Δεδομένων.

4.1 Πακέτο socialpserver

Το πακέτο αυτό περιλαμβάνει τα δομικά εργαλεία του project. Εδώ βρίσκονται οι κλάσεις που περιγράφουν τις δομές δεδομένων οι οποίες θα αντιπροσωπεύσουν τα αποτελέσματα, δηλαδή τα communities χρηστών καθώς και η main method. Εδώ επίσης έχουν τοποθετηθεί ως υπο-πακέτα τα υπολειπόμενα πακέτα της εφαρμογής: *.algorithmic* και *.dataio*
συγκεκριμένα εμπεριέχονται οι κλάσεις:

- ⋆ **SocialSPerver-main 4.1.1**
- ⋆ **Community 4.1.2**
- ⋆ **SetOfCommunities 4.1.3**



Σχήμα 4.1: socialPServer Package classDiagram

4.1.1 SocialPServer-main

Εδώ βρίσκεται και η μέθοδος **main** η οποία ταξινομεί τις λειτουργίες του προγράμματος. Σε πρώτο στάδιο η main καλείται να κάνει τις απαραίτητες αρχικοποιήσεις επομένως δημιουργεί την αρχική σύνδεση με την Βάση Δεδομένων την οποία θα χρησιμοποιήσουν οι ακόλουθες λειτουργίες και αρχικοποιεί τους *Loggers* ώστε να διαχειρίζονται με αποδοτικό τρόπο τα μηνύματα εξόδου. Στη συνέχεια δημιουργεί τους *DataLoaders* που θα αναλάβουν το πέρασμα των απαραίτητων δεδομένων και ανάλογα με την λειτουργία που πρέπει να εκτελεστεί καλεί τους αλγορίθμους να παράξουν κοινότητες. Τέλος διαχειρίζεται την έξοδο αυτών αποθηκεύοντας την παραγόμενη πληροφορία ή καλώντας την αξιολόγηση των αποτελεσμάτων.

Στην βασική αυτή κλάση ακόμα περιλαμβάνεται και η μέθοδος **socialUtils** η οποία παρέχει τις αρχικές πληροφορίες για τον κοινωνικό γράφο, πράγμα που βελτιώνει την λειτουργικότητα καθώς οι πληροφορίες αυτές είναι χρήσιμες τόσο για τον προγραμματιστή αλλά και για τον διαχειριστή της εκάστοτε υπηρεσίας, αφού εδώ γίνονται γνωστά χαρακτηριστικά όπως το πλήθος των χρηστών και των ακμών φιλίας.

4.1.2 Community

Σε αυτό το σημείο θα περιγραφεί τι πραγματικά σημαίνει κοινότητα χρηστών (community) για τον πρόγραμμά μας. Για να μπορεί ο socialPServer να αντιληφθεί και να χειριστεί μια κοινότητα έχει υλοποιηθεί η κλάση **Community** η οποία είναι η Δομή Δεδομένων που αντιπροσωπεύει μια τέτοια ομάδα χρηστών. Επομένως για κάθε παραγόμενη κοινότητα δημιουργούμε ένα νέο αντικείμενο αυτής της κλάσης και του περνάμε τις απαραίτητες πληροφορίες.

Ποιο αναλυτικά, η βάση μιας κοινότητας είναι ένα **java HashSet** που περιλαμβάνει userIDs. Ένα java Set είναι ιδανικό για να περιγράψει μια κοινότητα αφού, θέλουμε τα μέλη να είναι μοναδικά (*unique*) και δεν μας απασχολεί η σειρά με την οποία είναι αποθηκευμένα (*unordered*). Επίσης επιλέχθηκε συγκεκριμένα HashSet γιατί είναι η πιο αποδοτική δομή για τον χειρισμό τέτοιου είδους δεδομένων σε επίπεδο πόρων.
Με αυτήν την κλάση μπορούμε να προσθέσουμε τις επιπλέον λειτουργίες που χρειάζεται το πρόγραμμα καθώς και να προστατεύσουμε τη δομή από τη χρήση λειτουργιών των java Sets που δεν χρειάζονται στην προκειμένη περίπτωση και θα ήταν εν δυνάμει καταστροφικές. Ουσιαστικά έτσι δημιουργείται ένα 'υψηλότερο' επίπεδο χειρισμού δεδομένων αφού οι διάφορες μέθοδοι αλληλεπιδρούν με την κοινότητα μέσω αυτής της κλάσης.

Εμπεριέχεται μία global μεταβλητή:

```
private Set<String> community;
```

Αρχικά, με την δημιουργία ενός τέτοιου αντικειμένου ενεργοποιείται ο **constructor** που μπορεί να πάρει δύο μορφές, ανάλογα με το input που δίνεται.
Στην πρώτη περίπτωση μπορεί να δημιουργήσει κανείς ένα τέτοιο αντικείμενο χωρίς να δώσει κάποια παράμετρο γράφοντας:

```
Community comID = new Community();
```

όπου ο *constructor* αρχικοποιεί το *global community* με αυτόν τον τρόπο:

```
public Community() {
  this.community = new HashSet<>();
}
```

ή σε άλλη περίπτωση μπορεί από την πρώτη στιγμή να δώσει και το περιεχόμενο της δομής γράφοντας:

```
Community comID  = new Community(algorithm.out.comID);
```

όπου ο *constructor* αρχικοποιεί το *global community* κάνοντας:

```
public Community(Set<String> userCommunity) {
      this.community = new HashSet<>();
      for (String user : userCommunity) {
          community.add(user);
      }
}
```

Αφού αρχικοποιηθεί λοιπόν η δομή, ο δημιουργός του αντικειμένου πρέπει να μπορεί να εκτελέσει 'χαμηλότερου' επιπέδου λειτουργίες αλληλεπίδρασης. Υλοποιήθηκαν επομένως οι παρακάτω μέθοδοι:

addMember

πρόσθεση μέλους στην κοινότητα

```
community.add(member);
```

removeMember

αφαίρεση μέλους από την κοινότητα

```
community.remove(member);
```

getCommunityMembers

ανάκτηση μελών της κοινότητας

```
return community;
```

size

μέγεθος της κοινότητας

```
return community.size();
```

Τέλος, εδώ βρίσκονται κάποιες μέθοδοι που εμπλέκονται στην διαδικασία αξιολόγησης της παραγωγής κοινοτήτων δίνοντας σχετικές πληροφορίες, οι οποίες θα αναλυθούν στο κεφάλαιο **Evaluation 5.3** .

4.1.3 SetOfCommunities

Ο κάθε αλγόριθμος πρόκειται να παράξει ένα σύνολο communities επομένως προκύπτει η ανάγκη δημιουργίας μιας δομής που θα μπορεί να χειριστεί ένα τέτοιο σύνολο. Έτσι υλοποιήθηκε η κλάση **SetOfCommunities**.

Η δομή αυτή όπως προκύπτει και από το όνομά της βασίζεται σε ένα *global java HashSet* που περιλαμβάνει αντικείμενα τύπου *Community*.

```
private Set<Community> communities;
```

Και σε αυτήν την περίπτωση η δημιουργία ενός τέτοιου αντικειμένου μπορεί να πάρει δύο μορφές, αναλόγως με την μορφή του *constructor* που θα υιοθετήσουμε.

Στην πρώτη περίπτωση μπορεί να δημιουργήσει κανείς ένα τέτοιο αντικείμενο χωρίς να δώσει κάποια παράμετρο:

```
SetOfCommunities comSet = new SetOfCommunities();
```

όπου ο *constructor* αρχικοποιεί το *global Set<Community> communities* με αυτόν τον τρόπο:

```
public SetOfCommunities() {
    this.communities = new HashSet<>();
}
```

ή σε κάποια άλλη μπορεί από την πρώτη στιγμή να δώσει και το περιεχόμενο της:

```
SetOfCommunities comSet = new SetOfCommunities(algorithm.out);
```

όπου ο *constructor* αρχικοποιεί το *global Set<Community> communities* κάνοντας:

```
public SetOfCommunities(Set<Set<String>> communities) {
    this.communities = new HashSet<>();
    for (Set<String> community : communities) {
        this.communities.add(new Community(community));
    }
}
```

Όπως και προηγουμένως έτσι και σε αυτήν στην δομή μετά την αρχικοποίηση πρέπει να δίνονται στον προγραμματιστή κάποιες βασικές λειτουργίες για τον χειρισμό αυτού του Set. Υλοποιήθηκαν επομένως οι παρακάτω μέθοδοι:

addCommunity

πρόσθεση κοινότητας στο Set

```
communities.add(communityID);
```

removeCommunity

αφαίρεση κοινότητας από το Set

```
communities.remove(communityID);
```

getCommunities

ανάκτηση όλων των κοινοτήτων

```
return communities;
```

size

πλήθος των κοινοτήτων

```
return communities.size();
```

clear

finalize, free memory. Διαγραφή όλων των κοινοτήτων για ελευθέρωση της μνήμης και επίσης διαγραφή των υπόλοιπων δομών που περιέχουν δεδομένα

```
public void clear() {
    if (communities.size()>0) {
        communities.clear();
    }
    if (allCentroidFeatures.size()>0) {
        allCentroidFeatures.clear();
    }
}
```

Επίσης για μία συνολική ανασκόπηση των κοινοτήτων υπάρχει η μέθοδος:

communitiesInfo

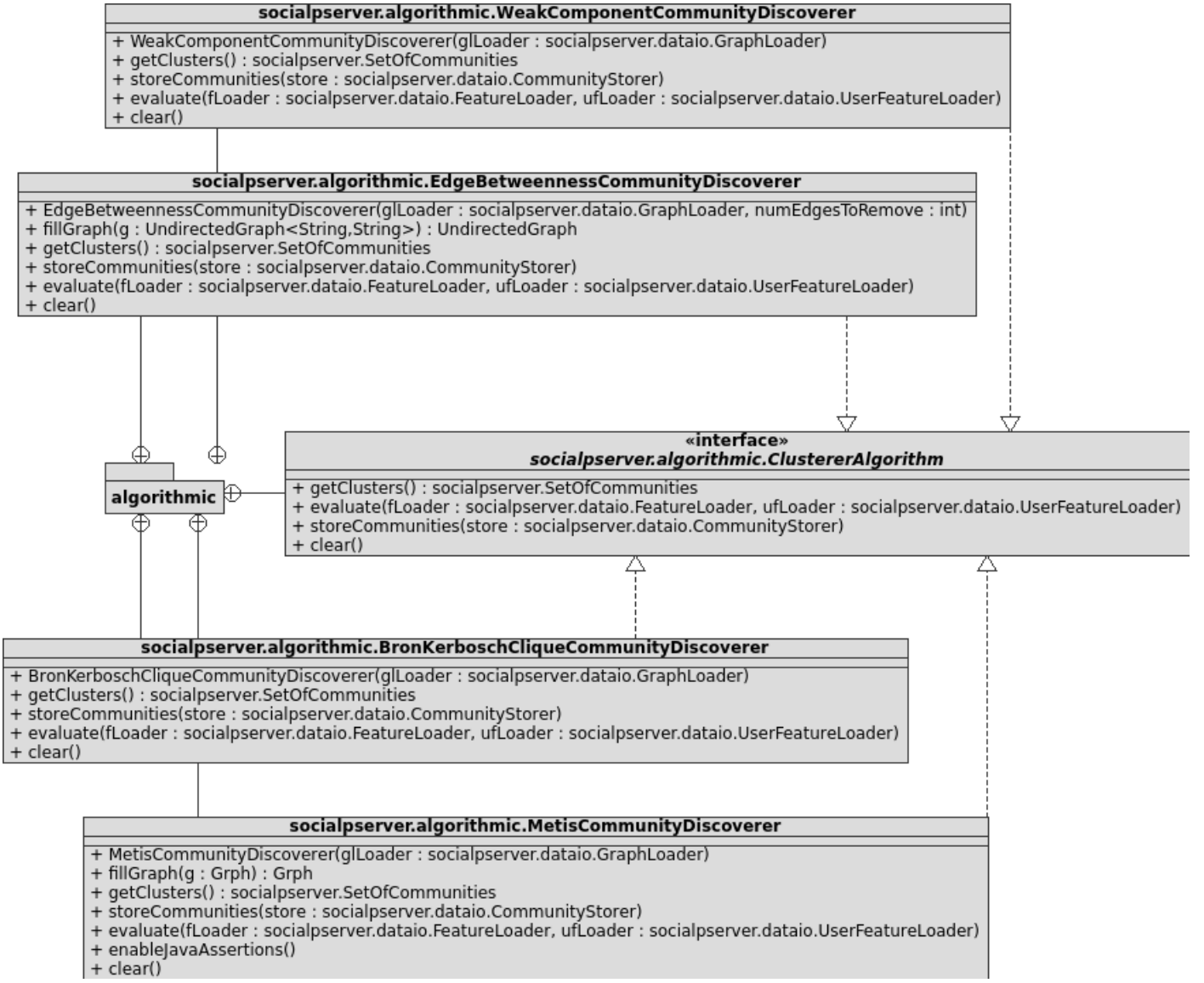
η οποία δίνει γενικές πληροφορίες όπως:
maxCommunitySize, minCommunitySize, averageCommunitySize, κτλ
(συνήθως καλείται αμέσως πριν την διαδικασία Evaluation αλλά μπορεί να καλεστεί και οποιαδήποτε άλλη στιγμή

Τέλος εδώ βρίσκονται κάποιες μέθοδοι που εμπλέκονται στην διαδικασία αξιολόγησης **Evaluation 5.3** , οι οποίες θα αναλυθούν στο σχετικό κεφάλαιο.

4.2 Πακέτο socialpserver.algorithmic

Εδώ περιγράφονται οι κλάσεις που αφορούν τους αλγορίθμους που χρησιμοποιούνται για την παραγωγή κοινοτήτων.

Όπως αναφέρθηκε και στο κεφάλαιο **2.4** η εξαγωγή κοινοτήτων χρηστών μπορεί να γίνει με πολλούς διαφορετικούς τρόπους. Χρησιμοποιήθηκε ένας αριθμός αλγορίθμων που που έχουν χρησιμότητα για την δικιά μας περίπτωση οι οποίοι είναι δανεισμένοι από java βιβλιοθήκες ανοιχτού κώδικα.



Σχήμα 4.2: .algorithmic Package classDiagram

4.2.1 interface ClustererAlgorithm

Αφού γίνεται χρήση αλγορίθμων που ανήκουν σε διαφορετικές βιβλιοθήκες, είναι φανερό πως πρέπει το πρόγραμμα να είναι σχεδιασμένο με τέτοιο τρόπο ώστε να υποστηρίζει την πρόσθεση και αφαίρεση αλγορίθμων και λειτουργιών. Πρέπει δηλαδή στην υλοποίηση να είναι ξεκάθαρα μεταξύ τους τα σημεία που είναι κοινά σε κάθε περίπτωση καθώς και ποιο σημείο έχει τον διαφοροποιημένο, για τον κάθε αλγόριθμο, κώδικα.

Για λόγους λοιπόν σαφήνειας και λειτουργικότητας δημιουργήθηκε το *interface: ClustererAlgorithm* το οποίο αποτελεί την σημαντικότερη κλάση στο πακέτο και είναι η βάση όλων των αλγορίθμων.
Πρόκειται για ένα πρότυπο το οποίο καθορίζει τις ελάχιστες μεθόδους που πρέπει να έχει μια κλάση του προγράμματος που αφορά αλγόριθμο.
Ποιο αναλυτικά:

imports

καθορίζονται τα ελάχιστα imports που πρέπει να έχει μια αλγοριθμική κλάση.

```
import socialpserver.SetOfCommunities;    //  Data Structure
import socialpserver.dataio.CommunityStorer;   // Data IO
import socialpserver.dataio.FeatureLoader;   // Data IO
import socialpserver.dataio.UserFeatureLoader;   // Data IO
```

μέθοδος getClusters()

Είναι υπεύθυνη για την εξαγωγή κοινοτήτων, θα καλεστεί τον εκάστοτε αλγόριθμο , θα δημιουργήσει το αντικείμενο SetOfCommunities, θα το επιστρέψει, κτλ

```
public SetOfCommunities getClusters();
```

μέθοδος evaluate()

Εδώ θα καλεστούν οι μέθοδοι που χρειάζονται για αξιολογηθεί η παραγωγή κοινοτήτων και κατ επέκταση ο αλγόριθμος. (Κεφάλαιο 5) Επίσης συγκεκριμενοποιείται πως για να καλεστεί η μέθοδος πρέπει να δοθούν ως παράμετροι οι dataLoaders οι οποίοι θα αναλάβουν να φορτώσουν όλα τα απαραίτητα για την διαδικασία δεδομένα

```
public void evaluate(FeatureLoader fLoader, UserFeatureLoader ufLoader);
```

μέθοδος storeCommunities()

καλείται για να πραγματοποιηθεί αποθήκευση των παραγόμενων κοινοτήτων. Χρειάζεται μια παράμετρο Storer η οποία καθορίζει τον στόχο αποθήκευσης (Βάση Δεδομένων, αρχείο, κτλ).

```
public void storeCommunities(CommunityStorer store);
```

μέθοδος clear()

Αναλαμβάνει να διαγράψει όσα στοιχεία δεν χρειάζονται πλέον, ώστε να ελευθερωθεί χώρος στην RAM. Συνήθως καλείται στην περίπτωση που κάποιος για πειραματικούς λόγους θέλει να τρέχει παραπάνω από έναν αλγορίθμους με μία εκτέλεση του προγράμματος και επομένως πρέπει να ελευθερωθεί η μνήμη που χρειάστηκε ο ένας πριν αρχίσει την διαδικασία ο επόμενος.

```
public void clear();
```

4.2.2 Ενδεικτικές Υλοποιήσεις Αλγορίθμων

Σε αυτό το σημείο παρουσιάζονται ενδεικτικές υλοποιήσεις των παραπάνω καθώς και κάποιες ακόμα μέθοδοι, οι οποίες με ελάχιστες ή καθόλου αλλαγές έχουν νόημα για κάθε αλγόριθμο:

η μέθοδος **fillGraph()**

Η συνάρτηση *fillGraph()* φροντίζει για την δημιουργία ενός γράφου που θα περιέχει τα δεδομένα στην μορφή που τα χρειάζεται σαν είσοδο ο κάθε αλγόριθμος. Όλες οι κλάσεις αλγορίθμων έχουν μια τέτοια μέθοδο, ο λόγος που δεν δηλώνεται στο interface είναι επειδή ο τύπος δεδομένων που δέχεται και επιστρέφει εξαρτάται κάθε φορά από τον τύπου του γράφο που χρειάζεται ο εκάστοτε αλγόριθμος.

Αρχικά ενημερώνεται ο Logger για την διαδικασία που θα ακολουθήσει και ανακτάτε την πληροφορία φιλίας μέσω του Loader:

```
socialPServerOutputLogger.info("filling from given loader");
Set<String[]> userAssociations = loader.getGraph();
```

Στη συνέχεια προσθέτονται στον γράφο οι χρήστες σαν κόμβοι και η φιλία σαν ακμή και τέλος επιστρέφεται ο γράφος.

για παράδειγμα στην περίπτωση του *SimpleGraph*:

```
for (String[] FriendsTable : userAssociations) {
    g.addVertex(FriendsTable[0]);
    g.addVertex(FriendsTable[1]);
    g.addEdge(FriendsTable[0], FriendsTable[1]);
}
return g;
```

και στην περίπτωση του *UndirectedGraph*:

```
Integer counter = 0;

for (String[] FriendsTable : userAssociations) {
    g.addEdge(counter.toString(), FriendsTable[0], FriendsTable[1]);
    counter++;
}

return g;
```

μέθοδος **getClusters()**

```
@Override
public SetOfCommunities getClusters() {
```

Αρχικά πρέπει να δημιουργηθεί ένας κοινωνικός γράφος στην μορφή που χρειάζεται ο κάθε αλγόριθμος για να λειτουργήσει.

στην περίπτωση των
EdgeBetweenness και *WeakComponent*

```
UndirectedGraph<String, String> g = new UndirectedSparseGraph<>();
```

στην περίπτωση του *BronKerboschClique*

```
SimpleGraph<String, DefaultEdge> g = new SimpleGraph<>(DefaultEdge.class);
```

στην περίπτωση του *Metis*

```
Grph g = new Grph();
```

Στην συνέχεια πρέπει ο γράφος αυτός να 'γεμίσει' με τους χρήστες και την πληροφορία φιλίας. Αυτό γίνεται καλώντας τη συνάρτηση *fillGraph()*.

```
fillGraph(g);
```

Πλέον ο κάθε αλγόριθμος είναι σε θέση να εντοπίσει κοινότητες χρηστών. Ενημερώνεται ο Logger για την διαδικασία που θα ακολουθήσει και καλείται η σχετική μέθοδος:

στην περίπτωση του *EdgeBetweenness*

```
algorithmOutputLogger.info("trying to find communities...");
EdgeBetweennessClusterer clusterer
          = new EdgeBetweennessClusterer(numEdgesToRemove);
town = new SetOfCommunities(clusterer.transform(g));
```

στην περίπτωση του *WeakComponent*

```
algorithmOutputLogger.info("trying to find communities...");
WeakComponentClusterer clusterer = new WeakComponentClusterer();
town = new SetOfCommunities(clusterer.transform(g));
```

στην περίπτωση του *BronKerboschClique*

```
algorithmOutputLogger.info("trying to find communities...");
BronKerboschCliqueFinder bk = new BronKerboschCliqueFinder(g);
Collection cliques = bk.getAllMaximalCliques();
```

στην περίπτωση του *Metis*

```
algorithmOutputLogger.info("trying to find communities...");
Gpmetis clusterer = new Gpmetis();
List<IntSet> clusterList = clusterer.compute(g, 20, new Random(5));
```

Όλοι οι αλγόριθμοι δεν επιστρέφουν τις κοινότητες στην ίδια μορφή δεδομένων επομένως σε αυτό το σημείο λαμβάνουν χώρα οι απαραίτητες ενέργειες για την μετατροπή των δεδομένων σε *SetOfCommunities*.

Τέλος δίνονται στον Logger βασικές πληροφορίες για τα παραγόμενα communities (πχ πλήθος) και επιστρέφεται το παραγόμενο SetOfCommunities.

```
  socialpserver.SocialPServer.algorithmOutputLogger.info
         ("EdgeBetweenness algoritm  ***found " + town.size() + " clusters***");
  return town;
}
```

μέθοδος **storeCommunities()**

Για την αποθήκευση κοινοτήτων έχοντας στην διάθεσή μας το αντικείμενο Storer, το οποίο απαιτείται σας παράμετρος, καλούμε την μέθοδό του .storeAll δίνοντας επίσης το αντικείμενο SetOfCommunities που έχει δημιουργηθεί και φορτωθεί με πληροφορία στην έξοδο του αλγορίθμου. Το μέσο της αποθήκευσης έχει καθοριστεί από αυτόν που δημιούργησε τον Storer.

```
  @Override
  public void storeCommunities(CommunityStorer store) {
      store.storeAll(socialComm);
  }
```

μέθοδος **clear()**

Η διαγραφή των εμπλεκόμενων δεδομένων γίνεται με το κάλεσμα των σχετικών μεθόδων των σχετικών δομών.

```
@Override
public void clear() {
    socialComm.clear();
}
```

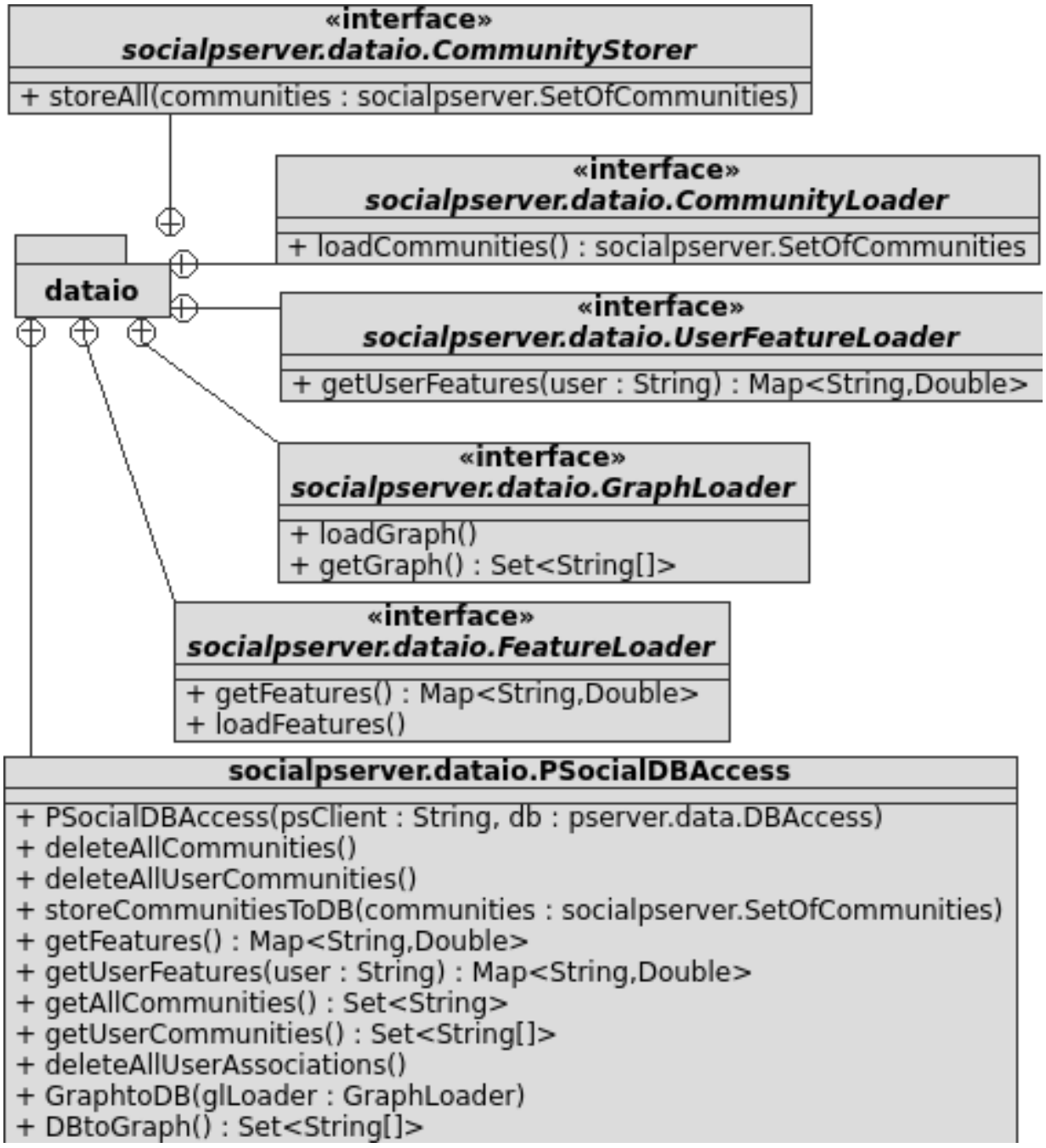
μέθοδος **evaluate()**

Στην αξιολόγηση αφού μελετούνται οι κοινότητες, καλούνται οι σχετικές μέθοδοι της δομής που τις αντιπροσωπεύει. Χρειάζεται βέβαια να τροφοδοτήσουμε με τα απαραίτητα δεδομένα, επομένως δίνουμε και ως παράμετρο τους σχετικούς dataLoaders.

```
@Override
public void evaluate(FeatureLoader fLoader, UserFeatureLoader ufLoader) {
    town.evaluate(fLoader, ufLoader);
}
```

4.3 Πακέτο socialpserver.dataio

Σε αυτό το πακέτο βρίσκονται όλες οι κλάσεις που αφορούν την Είσοδο και Έξοδο δεδομένων στο πρόγραμμα. Συγκεκριμένα περιέχονται κλάσεις σχετικές με *αποθήκευση δεδομένων, ανάκτηση δεδομένων,* επικοινωνία με τη *Βάση Δεδομένων* και οι *Loggers*.



Σχήμα 4.3: .dataIO Package classDiagram

4.3.1 Ανάκτηση - Αποθήκευση

Για λόγους επεκτασιμότητας, στο πρόγραμμα είναι υλοποιημένα τα interfaces **Loaders - Storers**. Δίνουν στον κάθε προγραμματιστής την δυνατότητα να ανακτά και να αποθηκεύει δεδομένα από και σε όποιο μέσο θέλει. Για κάθε τύπου πληροφορία έχει υλοποιηθεί το αντίστοιχο **πρότυπο** το οποίο περιγράφει πως πρέπει να είναι μια κλάση χειρισμού του ανάλογου τύπου δεδομένων, στην οποία μπορεί να βασιστεί κάποιος και να προσθέσει τις απαραίτητες για αυτόν μεθόδους.

Πρότυπα:

CommunityStorer

αποθηκεύει τις παραγόμενες κοινότητες, περιλαμβάνει τη μέθοδο:

```
public void storeAll(SetOfCommunities communities);
```

CommunityLoader

ανακτά τις κοινότητες που έχουν είδη παραχθεί (χρησιμοποιείται για το Evaluation), περιλαμβάνει τη μέθοδο:

```
public SetOfCommunities loadCommunities();
```

GraphLoader

ανακτά την πληροφορία φιλίας, περιλαμβάνει τις μεθόδους:

```
public void loadGraph();  \\ Load Graph in Memory
public Set<String[]> getGraph();   // return Graph
```

FeatureLoader

ανακτά όλα τα διαθέσιμα αντικείμενα της υπηρεσίας (χρησιμοποιείται για το Evaluation), περιλαμβάνει τις μεθόδους:

```
public Map<String, Double> getFeatures();
public void loadFeatures();
```

UserFeatureLoader

ανακτά τις προτιμήσεις κάθε χρήστη (αντικείμενα και αξιολογήσεις) (χρησιμοποιείται για το Evaluation), περιλαμβάνει τη μέθοδο:

```
public Map<String, Double> getUserFeatures(String user);
```

Παράδειγμα χρήσης:

Για ανάκτηση της κοινωνικής πληροφορίας από την βάση Δεδομένων δημιουργούμε ένα αντικείμενο της κλάσης:
GraphLoaderDB (implements GraphLoader)
στο οποίο δίνεται ως παράμετρος ένα αντικείμενο DBAccess για να διαθέτει επικοινωνία με την Βάση.
Αντίστοιχα στην περίπτωση ανάκτησης του κοινωνικού γράφου από αρχείο, δημιουργούμε ένα αντικείμενο της κλάσης:
GraphLoaderFile (implements GraphLoader)
στο οποίο ως παράμετρος δίνεται η σχετική διεύθυνση του αρχείου. Παρόμοια διαδικασία ακολουθείται για όλες τις I/O κλάσεις.

4.3.2 DBAccess

Στην κλάση **PSocialDBAccess** είναι υλοποιημένες οι απαραίτητοι μέθοδοι για την αλληλεπίδραση με την Βάση Δεδομένων. Ένας από τους στόχους του socialPServer είναι η επέκταση του Pserver, επομένως πρέπει να μπορεί να ανακτά και να αποθηκεύει δεδομένα που σχετίζονται με τη λειτουργία του από την Βάση Δεδομένων του PServer. Στην κλάση PSocialDBAccess βρίσκονται οι μέθοδοι που μεταφράζουν τις ανάγκες του προγράμματος για δεδομένα σε ερωτήσεις στην Βάση Δεδομένων του PServer.

Ποιο αναλυτικά, για πρόσβαση στην Βάση του PServer χρειάζεται ένα αντικείμενο *pserver.data.DBAccess*. Αυτό δίνεται ως παράμετρος στο πρόγραμμά μας και κατά την αρχικοποίηση δίνεται στην κλάση *PSocialDBAccess*. Επίσης σαν παράμετρος δίνεται το όνομα του *Πελάτη* (*Client* του pServer) για τον οποίο γίνεται η διαδικασία. Πελάτης είναι η κάθε υπηρεσία η οποία ζητάει προσωποποίηση και πρέπει να είναι γνωστοποιημένη για να πραγματοποιηθούν επιτυχώς τα queries στη βάση, αφού ταυτοχρόνως ο pServer ενδέχεται να εξυπηρετεί πολλούς πελάτες.

Κατά την υλοποίηση πρέπει να δοθεί προσοχή στις συνδέσεις που διατηρούνται ’ανοιχτές’ με την Βάση γιατί μπορεί να δημιουργηθεί φόρτος στον δίαυλο επικοινωνίας. Επομένως κατά την δημιουργία ενός αντικειμένου PSocialDBAccess : ο Constructor αναλαμβάνει να ανοίξει και αμέσως να κλείσει μια πρώτη σύνδεση:

```
dbAccess.connect();
dbAccess.disconnect();
```

ώστε όλες οι μέθοδοι που θα ακολουθήσουν απλά να επανασυνδέονται:

```
dbAccess.reconnect();
```

και συνολικά το πρόγραμμα να διατηρεί μία μόνο σύνδεση με τη Βάση Δεδομένων.

Για εγκατάσταση του socialPServer σε άλλο πρόγραμμα πρέπει να υλοποιηθεί η ανάλογη κλάση για επικοινωνία με τη Βάση (DBAccess) βασιζόμενη στις τωρινές μεθόδους.

4.3.3 Loggers

Καθ όλη την διάρκεια της λειτουργίας του, ο socialPServer δίνει μηνύματα τα οποία αποκαλύπτουν το στάδιο στο οποίο βρίσκεται η εκτέλεση, τα βήματα που έγιναν ή ακόμα και τα λάθη που προέκυψαν. Για την λειτουργική διαχείριση αυτών (output) χρησιμοποιούνται *java Loggers* και συγκεκριμένα η default βιβλιοθήκη της java *java.util.logging.Logger*.

Η διαδικασία Logging αποτελείται από τρία βασικά μέρη: *Logger, Handler, Formatter*. Έχουν υλοποιηθεί δύο σχετικές κλάσεις, η **LoggerInit** για τα πρώτα δύο και η **LoggerFormatter** για το τελευταίο.

Για την κάλυψη των αναγκών χρησιμοποιούνται δύο Loggers:
ο **socialPServerOutputLogger** ο οποίος θα αναλάβει όλο το output και το debugging του προγράμματος και ο
algorithmOutputLogger ο οποίος αφορά συγκεκριμένα την έξοδο των αλγορίθμων.

Στην κλάση **LoggerInit** η μέθοδος **initLogger** κάνει τις απαραίτητες αρχικοποιήσεις και ρυθμίσεις σχετικά με τους Loggers. Σε πρώτο στάδιο καθορίζεται η θεμιτή *ιεραρχία* των δυο Logger ώστε τα αλγοριθμικά μηνύματα να καταγράφονται αυτομάτως και στον socialPServerOutputLogger.

```
algorithmOutputLogger.setParent(socialPServerOutputLogger);
```

Στη συνέχεια καθορίζεται το *output target - Handler* που αντιπροσωπεύει το μέσω στο οποίο θα καταγράφονται τα μηνύματα. Εδώ προστέθηκαν δυο

Handlers, ένας για την εμφάνιση των μηνυμάτων στην *κονσόλα* και ένας για την καταγραφή σε *txt file* ώστε να είναι διαθέσιμα και μετά την εκτέλεση του προγράμματος.

```
ConsoleHandler consoleHandler = new ConsoleHandler();
fhAlgorithm = new FileHandler("./algorithmOutput.log");
fhSocialPServer = new FileHandler("./socialPServerOutput.log");
```

Για την ομαλή λειτουργία αφού χρησιμοποιούνται καινούργιοι consoleHandlers πρέπει να απενεργοποιηθεί ο default Handler της java.

```
LogManager.getLogManager().reset();
```

Επίσης καθορίζεται ποιος Handler αφορά ποιόν Logger:

```
algorithmOutputLogger.addHandler(consoleHandler); // add consoleHandler
algorithmOutputLogger.addHandler(fhAlgorithm);   // add FileHandler
socialPServerOutputLogger.addHandler(fhSocialPServer);   // add FileHandler
```

Κεφάλαιο 5

Αξιολόγηση παραγόμενων κοινοτήτων

Εδώ περιγράφεται η προσέγγιση και οι μέθοδοι που αφορούν την αξιολόγηση των παραγόμενων κοινοτήτων (Evaluation). Περιγράφονται σε ξεχωριστό κεφάλαιο αφού το Evaluation δεν αποτελεί λειτουργία του socialPServer αλλά εργαλείο του προγραμματιστή ώστε να μπορεί να συγκρίνει την απόδοση των αλγορίθμων.

Το πρόβλημα στην διαδικασίας αξιολόγησης είναι ότι δεν μπορεί να είναι ακριβής αν δεν υπάρξει ανάδραση από τους χρήστες. Αν δηλαδή δεν είναι γνωστό σε πιο βαθμό τα μέλη μιας κοινότητας έχουν όντως κοινά ενδιαφέροντα και προτιμήσεις. Για να ξεπεραστεί αυτό το πρόβλημα και να υπάρχουν στοιχεία αξιολόγησης, υλοποιήθηκαν οι παρακάτω μέθοδοι στις οποίες, έχοντας ένα DataSet που περιέχει ρεαλιστική κοινωνική πληροφορία μεταξύ των χρηστών αλλά και τις προτιμήσεις αυτών, ελέγχεται σε πιο βαθμό υπάρχει *ομοιότητα* στις αξιολογήσεις των αντικειμένων στα πλαίσια της κάθε κοινότητας. Καθώς επίσης και σε ποιο βαθμό υπάρχει *ανομοιότητα* στις αξιολογήσεις μεταξύ των διαφορετικών κοινοτήτων. Πρόκειται επομένως για **Εσωτερική Αξιολόγηση**. Ακολουθεί πιο αναλυτική περιγραφή της μεθόδου καθώς και παρουσίαση των DataSet που χρησιμοποιήθηκαν.

5.1 Δεδομένα (DataSets)

Όπως προαναφέρθηκε για να γίνει αξιολόγηση πρέπει να υπάρχουν πληροφορίες τόσο για την κοινωνική σχέση των χρηστών αλλά και για τις προτιμήσεις τους σε αντικείμενα. Για να είναι ρεαλιστική και να έχει αποτέλεσμα η διαδικασία δεν αρκεί να δημιουργηθεί ένα ενδεικτικό σύνολο δεδομένων παράγοντας τυχαίες συνδέσεις φιλίας και τυχαίες αξιολογήσεις αντικειμένων γιατί έτσι χάνεται το όλο νόημα της *κοινωνικής πληροφορίας* η οποία εμπεριέχεται σε πραγματικούς κοινωνικούς γράφους.

Επομένως πρέπει να χρησιμοποιηθούν ρεαλιστικά δεδομένα. Τέτοιου είδους δεδομένα σπανίζουν και συνήθως πληρώνονται αδρά, για αυτό και παρατηρείται η μανιωδώς συλλογή προσωπικών δεδομένων των χρηστών από τις μεγάλες εμπορικές πλατφόρμες που εμπεριέχουν κοινωνική δικτύωση.

Υπάρχουν όμως κάποια DataSets που είναι ανοιχτά για το κοινό, βοηθώντας την έρευνα, στα οποία είναι βασισμένες οι περισσότερες δημοσιεύσεις στον συγκεκριμένο τομέα. Στην παρούσα εργασία εντοπίσαμε τα ακόλουθα σύνολα δεδομένων:

Last.FM Dataset

Πρόκειται για ένα σύνολο δεδομένων που έχει εξαχθεί χρησιμοποιώντας την υπηρεσία μουσικής προσωποποίησης Last FM[27] το οποίο παρουσιάστηκε κατά τη διάρκεια του *2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems στο 5th ACM Conference on Recommender Systems*[28].
Περιλαμβάνει κοινωνική δικτύωση και αξιολογήσεις καλλιτεχνών, στοιχεία που το καθιστούν ιδανικό για την δικιά μας περίπτωση. Η έκδοση που χρησιμοποιήσαμε αποτελείται από 1892 χρήστες με 12717 ακμές φιλίας δηλαδή 25434 εγγραφές του τύπου: $undirected(user_i, user_j)$, όπου ο κάθε χρήστης έχει 13,443 φίλους κατά μέσο όρο. Επίσης δίνονται 92834 αξιολογήσεις καλλιτεχνών (αντικείμενα).
Η συγκεκριμένη έκδοση είναι δανεισμένη από το *Grouplens*[29].
Πρόκειται για *Undirected Graph*.

Flixster Dataset

Δεδομένα από το Flixster.com[30], μια πλατφόρμα για εύρεσης ταινιών. Προσφέρει την δυνατότητα αξιολόγησης ταινιών και μπορεί να εγκατασταθεί σε κοινωνικά δίκτυα από όπου και αντλήθηκε η πληροφορία

φιλίας. Επομένως δίνει social και rating πληροφορία. Η έκδοση που χρησιμοποιήσαμε είναι αυτή του Mohsen Jamali[31] η οποία περιλαμβάνει 7058820 εγγραφές φιλίας του τύπου: $undirected(user_i, user_j)$ και 8196078 αξιολογήσεις τενιών (ανικημένων).
Πρόκειται για *Undirected Graph*.

Epinions Dataset

Πρόκειται για μία διαδικτυακή υπηρεσία αξιολόγησης αντικειμένων, στην οποία κάποιος έχει την δυνατότητα να δηλώσει εμπιστοσύνη ή όχι για κάποιον άλλο χρήστη. Επομένως υπάρχει η πληροφορία trust-distrust η οποία συμβολίζεται με 1 ή -1 άλλα και τα ratings των αντικειμένων.
Αφού είναι trust και όχι social network πρόκειται για *Directed Graph*. [32]

Για την αξιολόγηση του socialPServer δοκιμάστηκαν δεδομένα του LastFM και του Flixster των οποίων η φύση ταιριάζει περισσότερο στην παρούσα εργασία.

5.2 Σχεδιασμός της μεθόδου Αξιολόγησης

Μέσω της πραγματικής χρήση του προγράμματος, κάποιος θα μπορούσε να πάρει το απαραίτητο feedback για την απόδοση του συστήματος, συλλέγοντας στατιστικά από την συμπεριφορά των χρηστών ανάλογα με τον κάθε αλγόριθμο. Θα μπορούσε επίσης με ένα λειτουργικό σύστημα συστάσεων, να κάνει προτάσεις στους χρήστες και ελέγξει το τι πραγματικά είδαν. Στο παρόν στάδιο όμως καθώς δεν έχει ολοκληρωθεί η λειτουργική ενσωμάτωση του socialPServer στον PServer, για να μπορεί να υπάρξει μιας μορφής αξιολόγηση και σύγκριση των αλγορίθμων ακολουθείται η τυπική προσέγγιση εσωτερικής αξιολόγησης clustering που προτείνεται όταν στόχος είναι η ομοιότητα μεταξύ των χρηστών σε επίπεδο αντικειμένων. Πιο συγκεκριμένα, η αξιολόγηση θεωρεί στόχο ενός ”καλού” αλγορίθμου την εύρεση κοινοτήτων με **υψηλή intra-cluster similarity** (οι χρήστες μέσα σε μία κοινότητα είναι *όμοιοι* σε επίπεδο αντικειμένων) και **χαμηλή inter-cluster similarity** (οι χρήστες που ανήκουν σε διαφορετικές κοινότητες είναι *ανόμοιοι* σε επίπεδο αντικειμένων). Πρόκειται για ένα κριτήριο *εσωτερικής* αξιολόγησης

και πρέπει να σημειωθεί πως καλή εσωτερική αξιολόγηση δεν σημαίνει απαραίτητα και καλή απόδοση του προγράμματος.
[33]

Η υλοποίηση μας είναι βασισμένη στην μέθοδο αξιολόγησης *Davies–Bouldin index* [34], η οποία παραλλάχθηκε ώστε να έχει νόημα για την περίπτωση.

Ποιο αναλυτικά, χρησιμοποιείται ένα μέτρο **ποιότητας συσταδοποίησης** για του οποίου τον υπολογισμό λαμβάνουν μέρος επιμέρους υπολογισμοί ομοιότητας (και όχι απόστασης όπως στην περίπτωση του Davies–Bouldin index). Το μέτρο ποιότητας υπολογίζεται ως εξής:

$$Quality = \frac{IntraSimmilarity}{InterSimmilarity}$$

όπου το Intra (Similarity) είναι η ομοιότητα που υπάρχει μεταξύ των μελών του ίδιου cluster και Inter (Similarity) η ομοιότητα που έχουν μεταξύ τους οι χρήστες διαφορετικών clusters. Όπως είναι προφανές στόχος του κάθε αλγορίθμου είναι να έχει υψηλό Intra και χαμηλό Inter Simmilarity. Για αυτό και η ποιότητα υπολογίζεται με τον λόγο αυτών των δύο ώστε να τηρούνται οι αναλογίες και να μπορούμε να συγκρίνουμε τους διαφορετικούς αλγορίθμους, με καλύτερο τον αλγόριθμο που έχει τον μεγαλύτερο λόγο (quality). Ακολουθεί η ποιο αναλυτική περιγραφή για το πως υπολογίζονται τα μέτρα Intra και Inter Simmilarity αφού πρώτα αναφερθούμε σε κάποιες βασικές δομικές έννοιες.

5.2.1 Αντιπροσωπευτικό σημείο συστάδας - centroid

Το πρόγραμμα μπορεί να εξυπηρετεί μεγάλο αριθμό χρηστών και τα αντικείμενα της υπηρεσίας μπορεί επίσης να είναι πάρα πολλά επομένως, η σύγκριση όλων των προφίλ χρηστών σε μία κοινότητα μεταξύ τους θα ήταν μια δαπανηρή διαδικασία τόσο σε επίπεδο χρόνου αλλά και πόρων. Για αυτό υπολογίζονται και χρησιμοποιούνται κεντροϊδή σε κάθε συστάδα **Centroids**. Για κάθε cluster δημιουργείται ένα ακόμα φανταστικό μέλος του οποίο το προφίλ (αξιολογήσεις) είναι το μέσο των αξιολογήσεων όλων των άλλων μελών. Αυτός ο εικονικός κόμβος είναι ουσιαστικά το κέντρο βάρους του γράφου των αντικειμένων - το σημείο που βρίσκεται στο κέντρο των αποστάσεων των προφίλ των χρηστών. Με αυτό τον τρόπο αντί να υπολογίζεται η ομοιότητα όλων των προφίλ των χρηστών μεταξύ τους υπολογίζεται η ομοιότητα του κάθε χρήστη με τον εικονικό χρήστη Centroid.

5.2.2 Μετρική ομοιότητας - cosine similarity

Υπάρχουν πολλές μετρικές που θα μπορούσαν να χρησιμοποιηθούν και να εκφράσουν την ομοιότητα δύο προφίλ χρηστών. Μία από τις πιο διαδεδομένες την οποία επιλέξαμε να χρησιμοποιήσουμε είναι η **Cosine similarity**. Η ομοιότητα Cosine συγκεκριμένα μετράει το συνημίτονο (cos) της γωνίας μεταξύ των δύο προφίλ. Από την φύση της παίρνει τιμές από -1 εώς 1 αλλά στις περιπτώσεις που οι αρνητικές τιμές δεν έχουν νόημα, όπως και εδώ, παίρνει τιμές από 0 ως 1.

Ο τύπος του υπολογισμού της:

$$cosineSimilarity(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

5.2.3 Μετρική ομοιότητας ζεύγους συστάδων - Intra Simmilarity

Ομοιότητα των μελών που βρίσκονται στο ίδιο cluster:

Έστω

C το πλήθος των clusters

M το πλήθος όλων των μελών όλων των cluster

m το πλήθος των μελών του κάθε cluster

x το κάθε Cluster

βασικές έννοιες:

ratio(x) cluster x size Ratio: βάρος του κάθε cluster στο συνολικό αποτέλεσμα

comp(x) cluster x Compuctness: εσωτερική ομοιότητα του κάθε cluster

$$Intra = \sum_{x=1}^{C} comp(x) * ratio(x)$$

$$ratio(x) = \frac{m_x}{M}$$

cos(i,j) cosine Simmilarity of memebers i - j

cent(x) centroid of cluster x

$$comp(x) = \frac{1}{m_x} * \sum_{i=1}^{m_x} cos(i, cent(x))$$

Για τον υπολογισμό του *Intra simmilarity* αθροίζουμε την εσωτερική ομοιότητα του κάθε cluster πολλαπλασιασμένη με το *Ratio* ώστε να επηρεάσει το τελικό αποτέλεσμα ανάλογα με το μέγεθός του. Για να βρούμε την εσωτερική ομοιότητα του κάθε cluster (*Compuctness*) αθροίζουμε την *cosine simmilarity* του κάθε μέλους με το centroid του cluster.

Επομένως προκύπτει:

$$InterSimmilarity = \frac{1}{M} \sum_{x=1}^{C} \sum_{i=1}^{m_x} cos(i, cent(x))$$

5.2.4 Inter Simmilarity

Ομοιότητα των μελών που **δεν** βρίσκονται στο ίδιο cluster:

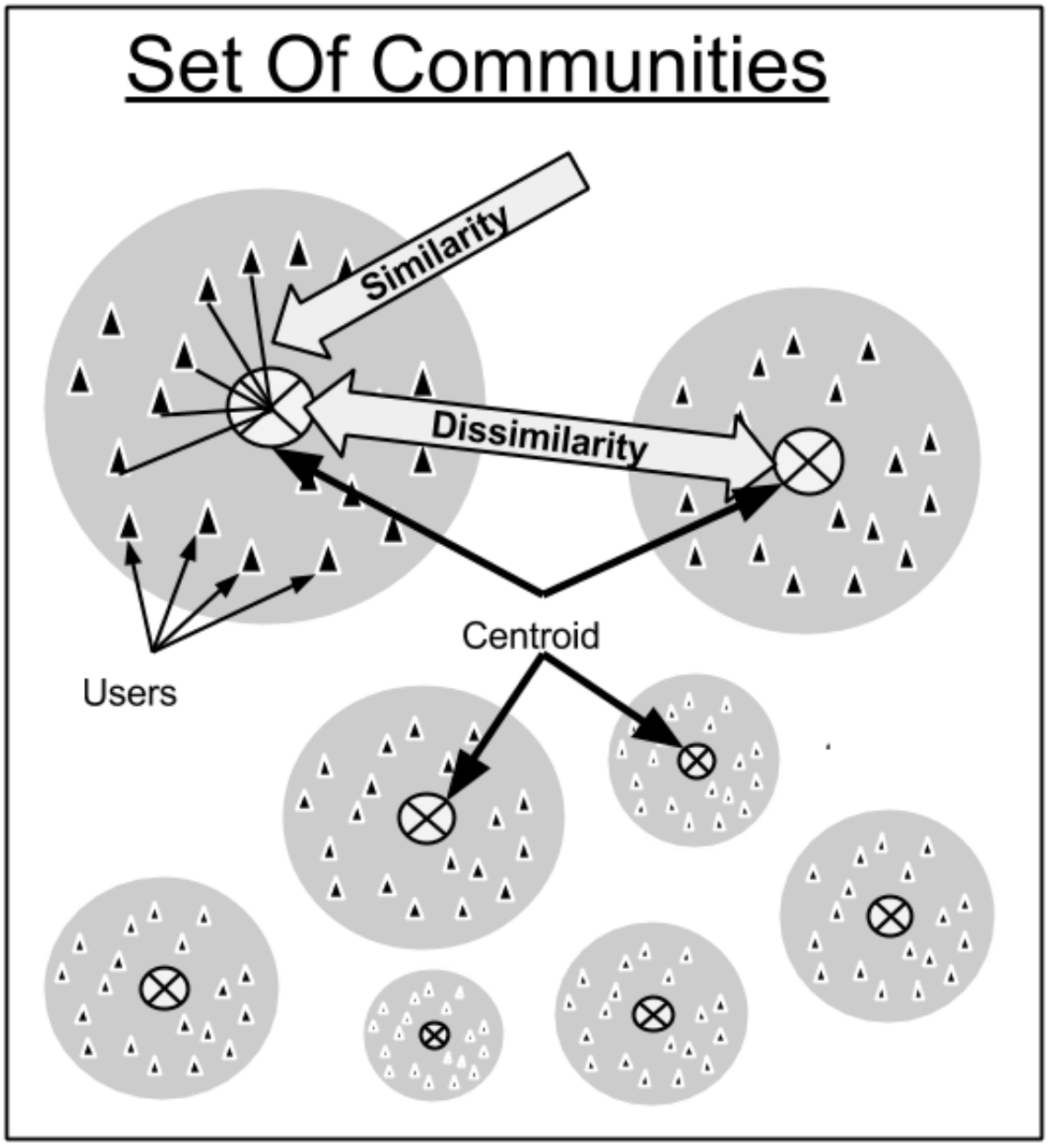
Με την ίδια λογική και σε αυτήν την περίπτωση αντί να συγκρίνουμε όλα τα μέλη των διαφορετικών κοινοτήτων μεταξύ τους, θα συγκρίνουμε τα *centroids* όλων των κοινοτήτων.

Επομένως:

$$InterSimmilarity = \sum_{i,j=0}^{C} cos(cent(i), cent(j))$$

5.3 Υλοποίηση της Αξιολόγησης

Οι μέθοδοι που σχετίζονται με την αξιολόγηση βρίσκονται κάθε φορά στην κλάση που αντιπροσωπεύει το αντικείμενο το οποίο θέλουμε να αξιολογήσουμε. Εφόσον αξιολογούνται αλγόριθμοι κάθε αλγόριθμος έχει μία μέθοδο **Evaluate**.
Ο κάθε αλγόριθμος περιέχει ένα αντικείμενο τύπου **SetOfCommunities** το οποίο αντιπροσωπεύει το σύνολο των παραγόμενων κοινοτήτων. Επομένως κάθε τέτοιου είδους αντικείμενο περιέχει επίσης μια μέθοδο **evaluation** την οποία καλεί ο εκάστοτε αλγόριθμος. Η μέθοδος evaluation() του κάθε SetOfCommunities αναλαμβάνει να υπολογίσει το **Quality** του Set. Κατά την διαδικασία θα χρειαστεί να υπολογιστεί η εσωτερική ομοιότητα της κάθε κοινότητας. Και σε αυτήν την περίπτωση, κάθε αντικείμενο τύπου Community



Σχήμα 5.1: αξιολόγηση κοινοτήτων

εμπεριέχει μια μέθοδο η οποία θα υπολογίσει το Intra Simmilarity της κάθε κοινότητας.

Κατά την διαδικασία της αξιολόγησης αρχικά υπολογίζονται κάποιες γενικές πληροφορίες για τις παραγόμενες κοινότητες όπως ο αριθμός των κοινοτήτων και το μέσο μέγεθος cluster και στην συνέχεια υπολογίζεται ο λόγος Quality που περιγράφτηκε παραπάνω.

Κεφάλαιο 6

Εφαρμογή Προγράμματος και Αλγορίθμων Σε Πραγματικά Δεδομένα - Συμπεράσματα

6.1 Εφαρμογή Σε Πραγματικά Δεδομένα

Προκείμενου να αποδειχθεί πως χρησιμοποιώντας την κοινωνική πληροφορία μπορούν να εξαχθούν συμπεράσματα για τις προτιμήσεις και την συμπεριφορά των χρηστών, δοκιμάσαμε το πρόγραμμά μας με δύο γνωστά DataSet. Η έρευνα είναι επιτυχής αν μέσω της μεθόδου αξιολόγησης που υλοποιήθηκε η μετρική *Total clustering scheme compactness* είναι μεγαλύτερη από την *Compactness of all centroids*, πράγμα που σημαίνει πως όντος τα μέλη μιας κοινότητας παρουσιάζουν ομοιότητες ενώ τα μέλη διαφορετικών κοινοτήτων ανομοιότητες.

$$ClustererQuality = \frac{InternalSimmilarity}{ExternalSimmilarity}$$

Ο κάθε αλγόριθμος έχει άλλη προσέγγιση στον τρόπο που εντοπίζει κοινότητες επομένως δεν έχει νόημα η σύγκρισή τους μόνο με βάση την μετρική **clusterer quality**. Πρέπει να είναι γνωστό ένα σύνολο γενικών χαρακτηριστικών που θα αναδείξουν την συμπεριφορά του κάθε ενός και θα μας επιτρέψουν να επιλέξουμε τον κατάλληλο για κάθε περίπτωση χρήσης.

6.1.1 Χαρακτηριστικά των Αλγορίθμων:

WeakComponent

Είναι στοχευμένος να εντοπίζει *κλίκες*. Δηλαδή ομάδες στις οποίες όλοι οι χρήστες ενώνονται με όλους και αυτό τον καθιστά κατάλληλο για περιπτώσεις που είναι επιθυμητές κοινότητες με πυκνές συνδέσεις στο εσωτερικό. Ομαδοποιεί μόνο όσους εμφανίζουν κλίκα και για αυτό μεγάλο μέρος τον χρηστών τελικά δεν ομαδοποιείται. Λόγο της φύσης του οι κοινότητες που εντοπίζει είναι σχετικά μικρού μεγέθους αλλά εκτελεί την διαδικασία σε αρκετά σύντομο χρόνο διάστημα.

BronKerbosch

Επίσης στοχευμένος να εντοπίζει κλίκες επομένως κοινότητες με πυκνές συνδέσεις. Κατάλληλος για την περίπτωσή μας αφού είναι σχεδιασμένος να εφαρμόζεται σε μη κατευθυνόμενους γράφους (undirected graphs) και να εντοπίζει *μέγιστες κλίκες*. Μεγίστη κλίκα είναι ένα σύνολο χρηστών στο οποίο όλοι έχουν σύνδεση μεταξύ τους και δεν μπορεί να προστεθεί άλλος κόμβος στην ομάδα και να διατηρηθεί αυτή η εσωτερική *πλήρης* συνδεσιμότητα. Το βασικό χαρακτηριστικό που τον διαφοροποιεί και τον καθιστά ευρέως χρησιμοποιούμενο είναι η δυνατότητά του να εντοπίζει επικαλυπτόμενες κοινότητες (overlapping communities). Ο κάθε χρήστης μπορεί να ανήκει σε περισσότερες από μία κοινότητες και επομένως *όλοι* οι χρήστες του γράφου ομαδοποιούνται.

EdgeBetweenness

Ιδανικός για Συστήματα Συστάσεων αφού εντοπίζει τους κεντρικούς διαύλους πληροφορίας στον γράφο. Επίσης κατάλληλος για Social Recommender Systems αφού έχει κατασκευαστεί σαν εργαλείο ανάλυσης κοινωνικών δικτύων που μετρά *κεντρικότητα*. Μια ακμή έχει υψηλό *EdgeBetweenness* όταν βρίσκεται στα *συντομότερα μονοπάτια (Shortest Paths)* πολλών χρηστών. Αυτό την κάνει να είναι πιο ”κεντρική” και επομένως αγωγός επιρροής. Απαιτείται βέβαια χρόνος και επεξεργαστική ισχύς για τον εντοπισμό των shortest paths.

EdgeBetwenness thresshold: Μεταβλητή που δίνεται ως παράμετρος και αντιπροσωπεύει τον αριθμό των ακμών που θα αφαιρεθούν σταδιακά από τον γράφο. Όσες περισσότερες ακμές αφαιρούνται, τόσο μικρότερα και πιο πυκνά τα clusters που θα προκύψουν.

Metis

Πακέτο αλγορίθμων για την κατάτμηση γράφων οι οποίοι λειτουργούν κάνοντας *Multilevel partitioning*. Αυτό σημαίνει πως αρχικά το μέγεθος του γράφου μειώνεται *συμπιέζοντας (coarsening)* τους συνδεδεμένους κόμβους και ακμές. Στην συνέχεια ο μικρότερος πλέον γράφος κατατμείται και στην συνέχεια επαναφέρεται στο κανονικό του μέγεθος. Κρίθηκε κατάλληλος για την περίπτωση των κοινωνικών δικτύων γιατί όταν ο γράφος βρίσκεται στην πιο συμπυκνωμένη του φάση (στην οποία και κατατμείται) έχουν συγχωνευτεί οι συνδεδεμένοι κόμβοι και επομένως αποκαλύπτονται οι διαφορετικές (μη συνδεδεμένες) κοινότητες.
Περιλαμβάνονται δύο βασικές λειτουργίες:
multilevel k-way partitioning (Ptype.kway) και
multilevel recursive bisectioning (Ptype.rb)
και στις δύο περιπτώσεις θα παραχθούν κοινότητες ίσου μεγέθους το πλήθος των οποίων δίνεται από τον χρήστη.
Στην περίπτωση του k-way ο γράφος χωρίζεται σε *k* ίσα μέρη ενώ στην περίπτωση του rb πραγματοποιείται αναδρομική διαδικασία κατά την οποία ο γράφος χωρίζεται κάθε φορά στο μισό και τερματίζει όταν υπάρξει ο επιθυμητός αριθμός κατατμήσεων.
Αφού οι εν λόγο αλγόριθμοι εφαρμόζονται στον συμπιεσμένο γράφο που προαναφέρθηκε η διαδικασία πραγματοποιείται πολύ πιο γρήγορα συγκριτικά με τους παραδοσιακούς αλγορίθμους οι οποίοι υπολογίζουν partitions κατευθείαν στον αρχικό (μεγαλύτερο) γράφο. [35] Οι αλγόριθμοι του Metis συνίστανται σε περιπτώσεις που είναι επιθυμητή η παραγωγή κοινοτήτων συγκεκριμένου μεγέθους αλλά και σε περιπτώσεις γράφων μεγάλου μεγέθους.
Το πακέτο δίνει την δυνατότητα λεπτομερειακής παραμετροποίησης και με την χρήση της μεθόδου αξιολόγησης μπορεί κανείς να πραγματοποιήσει fine tuning στην κάθε περίπτωση χρήσης και δεδομένων.

Παράμετροι του Metis με την σειρά που πρέπει να δύνονται:

Παράμ.	*Χαρακτηριστικά*	*Ερμηνεία*
g		ο γράφος
N		αριθμός επιθυμητών κατατμήσεων
ptype		καθορίζει τη μέθοδο που θα χρησιμοποιηθεί για τον υπολογισμό του k-way partitioning.
ctype		καθορίζει τη μέθοδο που θα χρησιμοποιηθεί για να να ταιριάξει τους κόμβους κατά το coarsening.
iptype	έχει ισχύ μόνο όταν ptype=rb.	καθορίζει τη μέθοδο που θα χρησιμοποιηθεί για τον υπολογισμό της αρχικής κατάτμησης του γράφου.
objtype	έχει ισχύ μόνο όταν ptype=kway.	καθορίζει τον στόχο που οι partitioning routines θα βελτιστοποιήσουν.
contig	Αν ο γράφος δεν είναι είναι συνδεδεμένος η παράμετρος αγνοείται. Έχει ισχύ μόνο όταν ptype=kway.	Καθορίζει ότι οι partitioning routines θα προσπαθήσουν να παράξουν κατατμήσεις που είναι συνεχόμενες.
minconn	έχει ισχύ μόνο όταν ptype=kway.	Καθορίζει ότι οι partitioning routines θα προσπαθήσουν να ελαχιστοποιήσουν τον μέγιστο βαθμό του subdomain γράφου, δηλ. του γράφου στον οποίο το κάθε partition είναι ένας κόμβος και οι ακμές συνδέουν τα subdomains που έχουν κοινό interface.
ufactor	Για ptype=rb, προεπιλεγμένη τιμή = 1 (i.e., load imbalance of 1.001). Για ptype=kway, προεπιλεγμένη τιμή = 30 (i.e., load imbalance of 1.03).	Καθορίζει το μέγηστο επιτρεπτό imbalance μεταξύ των κατατμήσεων. Μια τιμή x δηλώνει πως το επιτρεπτό load imbalance είναι $1+x/1000$. Για ptype=rb, το load imbalance μετριέται σαν την αναλογία: $2*max(left, right)/(left+right)$, όπου left και right είναι τα μεγέθη των αντίστοιχων partitions σε κάθε bisection (διχοτόμο γωνία). Για ptype=kway, το load imbalance μετριέται σαν την αναλογία: $max_i(pwgts[i])/avgpwgt$, όπου pwgts[i] είναι το βάρος του ith partition και avgpwgt είναι το άθροισμα του συνολικού βάρους των κόμβων, δια τον αριθμό τον ζητουμένων κατατμήσεων.
niter	Προεπιλογή = 10.	Καθορίζει τις φορές που θα τρέξουν οι αλγόριθμοι σε κάθε στάδιο της διαδικασίας uncoarsening.
ncuts	Προεπιλογή = 1.	Καθορίζει τον αριθμό των διαφορετικών κατατμήσεων που θα υπολογιστούν. Το τελικό partition είναι αυτό που επιτυγχάνει το καλύτερο edgecut ή βαθμό επικοινωνίας.
seed		Διαλέγει τον ”σπόρο” για την γεννήτρια τυχαίων αριθμών.
dbglvl		Επιλέγει το dbglvl.

6.1.2 Αλγόριθμοι Ομαδοποίησης (Clustering)

Χρησιμοποιώντας το Last.FM Dataset
Πλήθος Χρηστών: **1892**
Πλήθος κοινωνικών συνδέσεων: **12717**

Αλγόριθμος	*Edge Betweenness thresshold:5*	*Edge Betweenness thresshold:2*	*WeakComponent*	*BronKerbosch*
run time	89.521 sec	39.451 sec	0.037 sec	3.287 sec
number of Communities	21	20	19	11636
unclustered users	1834	1838	1843	0
max community size	7	7	7	10
min community size	2	2	2	2
average community size	2.762	2.7	2.579	3.5090237
Total clustering scheme compactness	0.5821527	0.5933855	0.6273950	12.303899
Compactness of all centroids	0.037443716	0.035636988	0.0356979	0.13966095
clusterer quality	15.547408	16.650833	17.575143	88.09834

$\surd$ Όλοι οι αλγόριθμοι παρουσιάζουν καλά αποτελέσματα σύμφωνα με την μετρική *clusterer quality* η οποία εκφράζει πως οι ομάδες χρηστών που εντοπίστηκαν παρουσιάζουν ομοιότητες περιεχομένου στο εσωτερικό τους και ανομοιότητες σε σχέση με τις άλλες κοινότητες. Επομένως αποδεικνύεται πως: *μην έχοντας καμία πληροφορία για τις προτιμήσεις των χρηστών μπορούν να εντοπιστούν ομάδες χρηστών με παρόμοιες προτιμήσεις.*

$\surd$ Τα αποτελέσματα του BronKerbosch είναι ενισχυμένα καθώς εντοπίζει επικαλυπτόμενες κοινότητες όπου ο κάθε χρήστης ανήκει σε πάνω από μια ομάδες. Με αυτόν τον τρόπο παράγεται μεγάλος αριθμός ομάδων στενά συνδεδεμένων μελών.

$\surd$ Στενά συνδεδεμένους χρήστες (κλίκες) επίσης εντοπίζουν οι EdgeBetweenness και WeakComponent οι οποίοι σχηματίζουν σχετικά μικρές ομάδες (2,5-3 κόμβοι). Με το χαρακτηριστικό όμως πως οι χρήστες του γράφου που δεν ανήκουν σε κλίκα αγνοούνται και μένουν εκτός κοινοτήτων (unclustered users).

√ Από το *run time* είναι εμφανές πως ο *WeakComponent* είναι ο αλγόριθμος που ολοκληρώνει πιο γρήγορα την διαδικασία. Ο χρόνος του EdgeBetweenness είναι αυξημένος αφού χρειάζεται χρόνος για τον εντοπισμό των shortest paths.

Χρησιμοποιώντας το Flixster Dataset

Πλήθος Χρηστών: **785926**

Πλήθος κοινωνικών συνδέσεων: **5861294**

Αλγόριθμος	*WeakComponent*
run time	17.27 sec
number of Communities	69
unclustered users	785754
max community size	4
min community size	2
average community size	2.4927535
Total clustering scheme compactness	0.20660953
Compactness of all centroids	0.007924251
clusterer quality	26.073067

√ Για δοκιμή του clustering στο *flixster dataset* χρησιμοποιήθηκε ο αλγόριθμος WeakComponent ο οποίος μπορεί να παράγει αποτελέσματα σε καλό χρόνο σε σχέση με τους EdgeBetweenness και BronKerbosch οι οποίοι σε περιπτώσεις τέτοιας πολυπλοκότητας (λόγο μεγέθους) χρειάζονται πολύ χρόνο για την ανάλυση του γράφου και την επεξεργασία των δεδομένων.

√ Ο αλγόριθμος WeakComponent εντοπίζει κοινότητες με ομοιότητες περιεχομένου αλλά πολύ λίγες σε σχέσει με τον αριθμό των χρηστών. Οι περισσότεροι χρήστες μένουν εκτός ομάδας. Επομένως για την εξόρυξη πληροφορίας από γράφους τέτοιου μεγέθους θα χρειαστεί κάποιας μορφής προ επεξεργασία των δεδομένων πιθανός με χρήση αλγορίθμων partitioning οι οποίοι θα μειώσουν την πολυπλοκότητα.

6.1.3 Αλγόριθμοι Κατάτμησης (Partitioning): Metis

Αλγόριθμος	*Metis1*	*Metis2*	*Metis3*	*Metis4*
run time	0.221 sec	0.282 sec	0.288 sec	108.963 sec
number of Communities	50	22	22	9000
unclustered users	0	0	0	0
max community size	38	88	87	89
min community size	35	83	85	0
average community size	37.84	86	86	87.32511
Total clustering scheme compactness	0.26465952	0.26450196	0.26567402	0.026727572
Compactness of all centroids	0.1481018	0.18055633	0.20126359	0.12443053
clusterer quality	1.7870108	1.4649277	1.3200302	0.21479915

Metis1: [Last FM]
(g, 50, Gpmetis.Ptype.kway, Gpmetis.Ctype.shem, Gpmetis.Iptype.random, Gpmetis.Objtype.cut, false, false, 30, 10, 1, new Random(5))
Metis2: [Last FM]
(g, 22, Gpmetis.Ptype.kway, Gpmetis.Ctype.shem, Gpmetis.Iptype.random, Gpmetis.Objtype.cut, false, false, 30, 10, 1, new Random(5))
Metis3: [Last FM]
(g, 22, Gpmetis.Ptype.rb, Gpmetis.Ctype.rm, Gpmetis.Iptype.grow, Gpmetis.Objtype.cut, false, false, 1, 10, 1, new Random(5))
Metis4: [flixster]
(g, 9000, Gpmetis.Ptype.kway, Gpmetis.Ctype.shem, Gpmetis.Iptype.random, Gpmetis.Objtype.cut, false, false, 30, 10, 1, new Random(5))

$\surd$ Τα αποτελέσματα της αξιολόγηση των αλγορίθμων partitioning δεν έχει νόημα να συγκριθούν με αυτά των αλγορίθμων clustering αφού έχουν διαφορετική στόχευση ως προς την επίλυση του προβλήματος.

$\surd$ Οι κατατμήσεις του γράφου περιλαμβάνουν μεγάλο αριθμό χρηστών και επομένως μεγάλο εύρος προτιμήσεων, σε αντίθεση με τους clustering αλγορίθμους που εντοπίζουν κλίκες. Για αυτό το λόγο και η μετρική *clusterer quality* φένεται να είναι χαμηλότερη στην περίπτωση του Metis.

$\surd$ Όλες οι κοινότητες είναι ίσου μεγέθους.

$\surd$ Ο αριθμός των κοινοτήτων καθορίζεται από τον χρήστη.

$\surd$ Οι δύο λειτουργίες του Metis, *k-way, recursive bisectioning* στην περίπτωση του LastFM DataSet έχουν όμοια απόδοση.

$\surd$ Και στα δύο DataSet οι αλγόριθμοι εντοπίζουν κοινότητες χρηστών με κάποιες ομοιότητες αλλά χωρίς να παρουσιάζονται ιδιαίτερες ανομοιότητες με τα άλλα σύνολα.

$\surd$ Ο Metis πραγματοποιεί την διαδικασία σε καλό χρόνο και στο *flixster DataSet*, αναλογικά με το μέγεθός του, αφού πρόκειται για *multilevel partioner*.

6.2 Συμπεράσματα

Στα πλαίσια αυτής της εργασίας πραγματοποιήθηκε μελέτη σχετικά με τα Συστήματα Συστάσεων και τον συνδυασμό τους με τα Κοινωνικά Δίκτυα. Διερευνήθηκε ο τρόπος με τον οποίο ένα τέτοιο σύστημα μπορεί να βελτιώσει την λειτουργία του, εξάγοντας γνώση από την κοινωνική διασύνδεση των χρηστών. Στην συνέχεια βάση αυτής της ανάλυσης αναπτύχθηκε ένα σύστημα εξαγωγής γνώσης από κοινωνικό γράφο, με τον κατάλληλο σχεδιασμό ώστε να αποτελέσει επέκταση ενός υπάρχοντος Recommendation System. Πρόκειται για τον PServer, ένα γενικευμένο σύστημα Προσωποποίησης το οποίο διαθέτει μηχανισμούς συστάσεων που ακολουθούν Content Based αλλά και Collaborative μεθόδους. Το σύστημά μας (socialPServer) πρόκειται για έναν μηχανισμό ο οποίος είναι σε θέση να χειρίζεται έναν κοινωνικό γάφο και να εφαρμόζει έναν αριθμό αλγορίθμων εντοπισμού κοινοτήτων χρηστών. Οι αλγόριθμοι μελετούν τις κοινωνικές συνδέσεις με διάφορους τρόπους και συμπεραίνουν ”κλίκες” στενά συνδεδεμένων χρηστών. Αυτή την κοινωνική πληροφορία πρόκειται να χρησιμοποιήσει ο PServer για να βελτιώσει την ακρίβεια των αποτελεσμάτων του. Σε μια ρεαλιστική κοινότητα τα μέλη μοιράζονται κοινά ”αντικείμενα”, για παράδειγμα κοινά συγγράμματα σε μία ακαδημαϊκή κοινότητα ή και κοινές αισθητικές προτιμήσεις σε έναν ευρύτερο φιλικό κύκλο. Επομένως για να επιτευχθεί η προσωποποίηση θα γίνουν υποθέσεις για τις προτιμήσεις των χρηστών ανάλογα με το κοινωνικό τους περιβάλλον.

Τέλος έγινε δοκιμή του προγράμματος σε πραγματικά δεδομένα με χρίση των DataSet: LastFM και Flixster. Για να αξιολογηθούν τα αποτελέσματα χρησιμοποιήθηκαν μέτρα ομοιότητας τα οποία υπολογίζουν κατά πόσον μοιάζουν τα προφίλ των χρηστών, δηλαδή σε ποιο βαθμό οι υπάρχουν κοινές προτιμήσεις αντικείμενων. Σε όλες τις περιπτώσεις των αλγορίθμων οι παραγόμενες κοινότητες εμφανίζουν ομοιότητα στα μέλη τις ίδιας κοινότητας και ανομοιότητα στα μέλη άλλων κοινοτήτων. Το γεγονός ότι μη ξέροντας τίποτα για τις προτιμήσεις μπορούν να εντοπιστούν σύνολα χρηστών τα οποία παρουσιάζουν ομοιότητες αποδεικνύει την δύναμη των πληροφοριών που κρύβει η κοινωνική δομή και το ενδιαφέρον για την ανάλυση των κοινωνικών δικτύων συνεχώς αυξάνεται.

Κεφάλαιο 7

Μελοντικές Επεκτάσεις - Προσωπικά Δεδομένα - Άδειες

7.1 Μελλοντικές Επεκτάσεις

Βελτιστωποίηση των συστάσεων

Στο κεφάλαιο 2 (Σχεδιασμός) όπου περιγράφεται η αρχιτεκτονική του PServer γίνεται λόγος για το *Recommendation Engine*, το ενδιάμεσο επίπεδο μεταξύ αυτού και της εφαρμογής. Σε αυτό το στάδιο επεξεργάζονται οι πληροφορίες που δίνει ο PServer ώστε να έχουν νόημα για την εκάστοτε υπηρεσία. Είδη έχει ξεκινήσει ο σχεδιασμός ενός τέτοιου μηχανισμού ο οποίος θα μπορεί να εκμεταλλευτεί πλήρως τις δυνατότητες του PServer συνδυάζοντας τις πληροφορίες που δίνονται σε κάθε λογικό επίπεδό (Personal, Community, Social).

Χρήση σε άλλες επιστήμες

Εμπνευσμένοι από το *Community structure in jazz*[36] όπου με collaborative προσέγγηση από τις πληροφορίες για συνδέσεις των jazz μουσικών και σχημάτων προκύπτουν κοινοτικές δομές, θα θέλαμε το πρόγραμμά μας να συμβάλει σε κοινωνικές και ανθρωπιστικές επιστήμες. Η χρήση του μπορεί να αναδείξει νέες κοινωνικές ομάδες και ρεύματα, συνεπώς μπορεί να συμβάλει στην καλύτερη κατανόηση του κοινωνικού μας περιβάλλοντος και κατ επέκταση στην καλύτερη διαχείριση.

Επικαλυπτόμενες Κοινότητες

Μελετώντας την σχετική βιβλιογραφία, η πιο πλούσια και ρεαλιστική προσέγγιση όσον αφορά τις κοινότητες συνδεδεμένων μελών που εγείρει το ενδιαφέρον μας και ανοίγει νέους ορίζοντες για έρευνα είναι

το άρθρο *Uncovering the overlapping community structure of complex networks in nature and society*[37]. Οι συγγραφείς του ερμηνεύουν τα δικτυακά συστήματα στη φύση και την κοινωνία ως την συνύπαρξη των μικρότερων κοινοτήτων που τις αποτελούν. Αναγνωρίζοντας πως τέτοια σύνολα εμφανίζουν επικαλυπτόμενες κοινοτικές δομές, εισάγουν μετρικές και μεθόδους για την περαιτέρω μελέτη τους, προσπαθώντας να αποκαλύψουν την πραγματική τους φύση και ξεπερνώντας τις μέχρι τώρα ντετερμινιστικές μεθόδους. Μακροπρόθεσμος στόχος μας είναι να επεκταθεί η παρούσα εργασία σε ένα σύστημα που θα εκμεταλλεύεται τις πληροφορίες που αντλούνται από ένα περίπλοκο σύστημα με επικαλυπτόμενες κοινότητες ώστε να εξάγει γνώσει από αυτό.

7.2 Προσωπικά Δεδομένα

Οι τεχνολογίες προσωποποίησης χρησιμοποιούνται πλέον σε πληθώρα διαδικτυακών εφαρμογών και υπηρεσιών. Αποτελούν ανεκτίμητο εργαλείο αλλά ταυτόχρονα φέρνουν σημαντικές δεοντολογικές συγκρούσεις στον επιστημονικό κόσμο. Αυτό συμβαίνει διότι κατά την δημιουργία των προφίλ χρηστών που προαναφέρθηκαν, θίγονται ευαίσθητα θέματα και εμφανίζονται ηθικά διλήμματα τα οποία σχετίζονται με την **παραβίαση προσωπικών δεδομένων**. Η διαδικασία συλλογής των δεδομένων μπορεί να πραγματοποιείται *άμεσα*, όπως στις περιπτώσει που ο ίδιος ο χρήστης καταχωρεί στοιχεία για αυτόν, είτε *έμμεσα* όπως στις περιπτώσεις που βγαίνουν συμπεράσματα από την αλληλεπίδρασή του με το σύστημα.[3]
Και στις δυο περιπτώσεις ο χρήστης χάνει την ανωνυμία του και επομένως πρέπει να γνωρίζει πως τα προσωπικά του δεδομένα καταγράφονται και χρησιμοποιούνται από την υπηρεσία. Ακόμα και στις περιπτώσεις που ο ίδιος έχει επιτρέψει στην υπηρεσία την καταγραφή δεδομένων, μέσω της χρήσης Cookies, τέτοιου είδους πληροφορίες μπορούν να ανακατευθυνθούν σε άλλους διαδικτυακούς τόπους ώστε να υποκλαπούν.

Σε κάθε περίπτωση εφαρμογής τέτοιου είδους τεχνολογιών πρέπει από την αρχή να γίνεται σαφές στον χρήστη πως οι κινήσεις του καταγράφονται ώστε να φροντίζει μόνος του για την επιλογή των δεδομένων που θα γνωστοποιηθούν. Αυτό βέβαια προϋποθέτει πως και τα άτομα που διευθύνουν

την υπηρεσία είναι άξια εμπιστοσύνης. Ελπίζουμε και επιδιώκουμε τα αποτελέσματα αυτής της εργασίας να μην χρησιμοποιηθούν με αρνητική σκοπιμότητα άλλα για να επηρεάσουν θετικά την ανθρώπινη και την κοινωνική ζωή.

7.3 Άδειες

Στα πλαίσια της παρούσας πτυχιακής χρησιμοποιήθηκαν τα παρακάτω εργαλεία κατασκευασμένα από άλλους προγραμματιστές τα οποία έχουν εκδοθεί κάτω από άδειες ανοιχτού κώδικά.

project	**Licence**	**Source**
WeakComponent	BSD open-source license	JUNG Software Library `http://jung.sourceforge.net/license.txt`
EdgeBetweenness	BSD open-source license	JUNG Software Library `http://jung.sourceforge.net/license.txt`
BronKerbosch	EPL (Eclipse) and LGPL	jgrapht Java graph Library `https://github.com/jgrapht/jgrapht/wiki/Relicensing`
Metis	Apache License, Version 2.0	Karypis George `https://www.apache.org/licenses/LICENSE-2.0`
GPmetis	LGPL Version 3	Grph - graph library `http://www.i3s.unice.fr/~hogie/grph/?page=License`
betweenness figure 2.1	GNU 1.2, CC 3.0, CC 2.5	Claudio Rocchini(2007) - Wiki `https://en.wikipedia.org/wiki/File:Graph_betweenness.svg`
PServer	Apache License, Version 2.0	Dimokritos - SciFy `http://pserver-project.org/en/content/pservers-source-code`

Βιβλιογραφία

[1] Maurice D Mulvenna, Sarabjot S Anand, and Alex G Büchner. Personalization on the net using web mining: introduction. *Communications of the ACM*, 43(8):122–125, 2000.

[2] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[3] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Transactions on Internet Technology (TOIT)*, 3(1): 1–27, 2003.

[4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99.

[5] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[6] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[7] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[8] Leonard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741. URL http://www.math.dartmouth.edu/~{}euler/pages/E053.html.

[9] KannaAl Falahi, Nikolaos Mavridis, and Yacine Atif. Social networks and recommender systems: A world of current and future synergies. *Computational Social Networks*, pages 445–465, 2012. doi: 10.1007/978-1-4471-4048-1_18. URL http://dx.doi.org/10.1007/978-1-4471-4048-1_18.

[10] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. pages 287–296, 2011. doi: 10.1145/1935826.1935877. URL http://doi.acm.org/10.1145/1935826.1935877.

[11] Nina Mishra, Robert Schreiber, Isabelle Stanton, and RobertE. Tarjan. Clustering social networks. In Anthony Bonato and FanR.K. Chung, editors, *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 56–67. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77003-9. doi: 10.1007/978-3-540-77004-6_5. URL http://dx.doi.org/10.1007/978-3-540-77004-6_5.

[12] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004. doi: 10.1103/PhysRevE.69.026113. URL http://link.aps.org/doi/10.1103/PhysRevE.69.026113.

[13] Igor Kabiljo. Social graph clustering.

[14] Scott White. Weak component clusterer, a tool taken from jung2 2.0 api, 2009. URL http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/cluster/WeakComponentClusterer.html.

[15] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[16] Claudio Rocchini. Graph betweenness img, hue scale representing node betweenness on a graph, April 2007. URL https://en.wikipedia.org/wiki/File:Graph_betweenness.svg.

[17] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*,

99(12):7821–7826, 2002. doi: 10.1073/pnas.122653799. URL http://www.pnas.org/content/99/12/7821.abstract.

[18] Tom Nelson Scott White. Edge betweenness clusterer, a tool taken from jung2 2.0 api, 2009. URL http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/cluster/EdgeBetweennessClusterer.html.

[19] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973. ISSN 0001-0782. doi: 10.1145/362342.362367. URL http://doi.acm.org/10.1145/362342.362367.

[20] R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure. *Journal of molecular biology*, 279(1):287–302, May 1998. ISSN 0022-2836. doi: 10.1006/jmbi.1998.1689. URL http://dx.doi.org/10.1006/jmbi.1998.1689.

[21] Ewgenij Proschak. Bronkerbosch clique finder, tool taken from jgrapht java graph library. URL http://jgrapht.org/javadoc/org/jgrapht/alg/BronKerboschCliqueFinder.html.

[22] Karypis George. set of serial programs for partitioning graphs. URL http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.

[23] Karypis George. Metis manual, version 5.1.x. *University of Minnesota, Department of Computer Science and Engineering*, 2013.

[24] Grph. Gpmetis, tool taken from grph high performance graph library for java. URL http://www.i3s.unice.fr/~hogie/grph/javadoc/grph/algo/partitionning/metis/Gpmetis.html.

[25] National Center for Scientific Research ”Demokritos” and SciFY. Pserver project, . URL http://pserver-project.org/en.

[26] National Center for Scientific Research ”Demokritos” and SciFY. User’s guide for pserver, . URL pserver-project.org/sites/default/files/PServerUsersGuide.pdf.

[27] Last FM. Last.fm online music system. URL http://www.lastfm.com.

[28] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

[29] grouplens. grouplens lastfm dataset. URL http://grouplens.org/datasets/hetrec-2011/.

[30] flixster.com. Flixster is a social movie site. URL http://www.flixster.com/.

[31] Mohsen Jamali sja25@sfu.ca. Flixster, 2010. URL http://www.cs.sfu.ca/~sja25/personal/datasets/.

[32] Paolo Massa and Paolo Avesani. Trust-aware bootstrapping of recommender systems. In *ECAI Workshop on Recommender Systems*, pages 29–33. Citeseer, 2006.

[33] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1, chapter: Flat clustering-Evaluation of clustering. Cambridge University Press Cambridge, 2008.

[34] David L. Davies and Donald W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227, April 1979. ISSN 0162-8828. doi: 10.1109/TPAMI.1979.4766909.

[35] Pekka Miettinen, Mikko Honkala, and Janne Roos. *Using metis and hmetis algorithms in circuit partitioning*. Helsinki University of Technology, 2006.

[36] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[37] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.