Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Γραφοθεωρητικές Μέθοδοι Κοινωνικής Σύστασης** **Social Recommendation: A Graph – Based Approach**
Ονοματεπώνυμο Φοιτητή	**Χαράλαμπος Ρενιέρης**
Πατρώνυμο	**Παναγιώτης**
Αριθμός Μητρώου	**ΜΠΠΛ 14073**
Επιβλέπων	**Γεώργιος Τσιχριντζής, Καθηγητής**
Τριμελής επιτροπή	**καθ. κ. Τσιχριντζής Γ. – Επίκ. Καθ. κ. Αλέπης Ε. – Διδάκτωρ κ. Σωτηρόπουλος Δ.**

Πίνακας Περιεχομένων

Λίστα Γραφημάτων

Λίστα Εξισώσεων

Λίστα Εικόνων

Λίστα Πινάκων

Περίληψη

Ο σκοπός της διπλωματικής εργασίας είναι η κατασκευή αλγορίθμου ο οποίος θα μπορεί να βρίσκει την πιθανή βαθμολογία κάποιου χρήστη μιας ιστοσελίδας αξιολόγησης προϊόντων, βάση διάφορων παραγόντων όπως τις πραγματικές βαθμολογίες των φίλων του και τον βαθμό συσχέτισης μεταξύ χρηστών ανεξάρτητα αν έχουν σχέση φιλίας. Με τον τρόπο αυτό μπορούμε να εξετάσουμε τις προτιμήσεις κάποιου χρήστη, να κατηγοριοποιήσουμε τους χρήστες σε μεγάλες ομάδες χρηστών με υψηλό βαθμό συσχέτισης καθώς και να κάνουμε πιο σωστές προτάσεις προϊόντων ή δραστηριοτήτων στους χρήστες.

Συγκεκριμένα, χρειάστηκε η κατασκευή ενός εργαλείου που να παίρνει τα δεδομένα που χρειαζόμαστε από την ιστοσελίδα που επιλέξαμε και στη συνέχεια να τα καταχωρίζει στη βάση δεδομένων. Η ιστοσελίδα που επιλέξαμε και πληροί τις προϋποθέσεις της ερευνάς μας είναι η https://www.goodreads.com/ . Η σελίδα αυτή φιλοξενεί ένα παγκόσμιο κατάλογο βιβλίων. Οι εγγεγραμμένοι χρήστες της μπορούν να επιλέξουν τα βιβλία που έχουν διαβάσει και να τα βαθμολογήσουν, να επιλέξουν τα βιβλία που θα ήθελαν να διαβάσουν και μπορούν να γίνουν φίλοι με άλλους χρήστες. Καλύπτοντας λοιπόν τα τρία βασικά στοιχεία που χρειάζονταν για την εκπόνηση της έρευνας αυτής, προϊόν, βαθμολογία και σχέση φιλίας, επιλέχθηκε η σελίδα αυτή.

Η παρούσα εργασία αρχικά παρουσιάζει τον τρόπο που χρησιμοποιήθηκε ώστε να πάρουμε τα δεδομένα που χρειαζόμαστε για την έρευνα και την εφαρμογή δύο αλγορίθμων πάνω στα δεδομένα αυτά.

Abstract

The scope of this thesis is the development of an algorithm which can find the possible rating that a user of a rating website would give to a product, depending on various factors such as the real given ratings of the user's friends and the correlation value between the users whether they are friends or not. With this method, we can look into the preferences of a user, to group the user into bigger users' groups with a higher correlation value as also to do more correct product or activities suggestions to users.

Specifically, the development of a tool which can provide us with the data that we would need from the website that we have selected and then submit the data into our database, was necessary. The website that we have selected and fulfills the requirements of our research is https://www.goodreads.com/ . This website hosts a global books catalogue. The registered users of this website can select the books that they have read and rate them, they can select the books that they would like to read and they can become friends with other users. So, by covering the three main requirements needed for the realization of this research, list of products, ratings and friendship, this website was selected.

The present thesis, initially, presents the way – tool used in order to collect the data needed for the research and the development and application of two algorithms to the collected data.

1. Εισαγωγή

Η ανάπτυξη του διαδικτύου, τόσο στον αριθμό των χρηστών που το χρησιμοποιούν καθημερινά, όσο και στον αριθμό των παρεχόμενων υπηρεσιών, έχει δημιουργήσει ένα τεράστιο όγκο διαθέσιμων, ψηφιακών πληροφοριών και δεδομένων. Το πρόβλημα που δημιουργεί ο όγκος αυτός είναι ο χρόνος που θα σπαταλήσει κάποιος για να αναζητήσει την πληροφορία ή τα δεδομένα που θέλει. Οι σύγχρονες μηχανές αναζήτησης (Google, Bing, Yahoo κ.α.) δίνουν την αρχική λύση, μειώνοντας το χρόνο που θα δαπανήσει ο χρήστης για την ανεύρεση της πληροφορίας που αναζητά, με τα φίλτρα και τους αλγορίθμους αναζήτησης που έχουν υλοποιήσει και προσφέρουν.

Παρ 'όλα αυτά, οι λύσεις αυτές εστιάζουν στην αναζήτηση συγκεκριμένων πληροφοριών με βάση τις λέξεις κλειδιά που τους παρέχει ο χρήστης, τις προηγούμενες αναζητήσεις του ή τις σελίδες που επισκέφτηκε πρόσφατα. Έτσι η απαίτηση για συστήματα πιο προσωποποιημένων φίλτρων στις προσφερόμενες πληροφορίες σε κάθε χρήστη, άρχισε να αυξάνεται. Εδώ λοιπόν, εμφανίζονται τα συστήματα σύστασης (Recommender systems) τα οποία είναι συστήματα φιλτραρίσματος πληροφοριών και προσφέρουν αυτό το προσωποποιημένο και πιο στοχευμένο φιλτράρισμα των πληροφοριών ανάλογα με τις προτιμήσεις, τα ενδιαφέροντα, τα πρότυπα που δημιουργούνται από την παρατήρηση της συμπεριφοράς του χρήστη.

Τα συστήματα σύστασης έχουν αποδειχτεί πολύ ωφέλιμα και για τους χρήστες και για τους παρόχους κάποιας υπηρεσίας. Σε περιβάλλον ηλεκτρονικού εμπορίου, μειώνουν την επικοινωνία με το server καθώς η αναζήτηση και επιλογή κάποιου προϊόντος είναι συντομότερη, έτσι οι πάροχοι

της υπηρεσίας εξοικονομούν πόρους ενώ ταυτόχρονα κάνοντας τη διαδικασία επιλογής του προϊόντος πιο αποτελεσματική, βελτιώνουν και τις πωλήσεις τους. Για τους χρήστες, βελτιώνεται η διαδικασία και η ποιότητα επιλογής. Σε περιβάλλον ηλεκτρονικής βιβλιοθήκης, επιτρέπει στους χρήστες να βρίσκουν σχετικές και αξιόπιστες πληροφορίες πέρα από τις αναζητήσεις τους.

- Προσέγγιση με βάση το περιεχόμενο

Η προσέγγιση με βάση το περιεχόμενο έχει σαν στόχο να αναλύσει και να αναγνωρίσει τα κοινά χαρακτηριστικά των αντικειμένων που έχουν δεχτεί μια καλή κριτική από κάποιους χρήστες και στη συνέχεια να προτείνουν στους χρήστες αυτούς, καινούρια αντικείμενα τα οποία έχουν αυτά τα χαρακτηριστικά. Στα συστήματα που εφαρμόζεται αυτή η προσέγγιση είναι κυρίως τα συστήματα τα οποία προσφέρουν αντικείμενα με πλούσιο περιεχόμενο όπως κείμενα ή ταινίες.

Σύμφωνα με αυτή την τεχνική, δημιουργούμε και ενημερώνουμε μετά από κάθε βαθμολογία που δίνει ο χρήστης, ένα προσωπικό προφίλ. Το προφίλ ενημερώνεται προσθέτοντας στο βάρος του, το βάρος του προφίλ που έχει δημιουργήσει το σύστημα για το προϊόν. Έτσι το σύστημα μπορεί να προτείνει καινούρια προϊόντα των οποίων η βαθμολογία του προφίλ τους να είναι παρόμοια με τη βαθμολογία του προφίλ του χρήστη.

Τα συστήματα σύστασης τα οποία βασίζονται μόνο στο περιεχόμενο, έχουν συνήθως προβλήματα με την ανάλυση του περιεχομένου του αντικειμένου, δηλαδή δεν έχουν αρκετά στοιχεία για την ανάλυση του αντικειμένου ή το ακριβώς αντίθετο, έχουν πάρα πολλές λεπτομέρειες για ένα αντικείμενο κάτι που οδηγεί στην υπερεξειδίκευσή του.

Το πρώτο πρόβλημα, της έλλειψης επαρκών πληροφοριών, μπορεί να οφείλεται σε διάφορους λόγους όπως περιορισμός ή απαγόρευση στη χρήση προσωπικών πληροφοριών του χρήστη έτσι δεν μπορεί να διαμορφωθεί ένα αντιπροσωπευτικό προφίλ για το χρήστη και η πρόταση θα είναι τυχαία. Επιπροσθέτως, οι πληροφορίες του αντικειμένου θα μπορούσαν να είναι και αυτές ελλιπείς, ειδικά σε αντικείμενα τα οποία περιλαμβάνονται στους κανονισμούς περί πνευματικής ιδιοκτησίας, όπως ταινίες, εικόνες κ.α. Από την άλλη πλευρά το σύστημα υπερεξειδικεύοντας την πληροφορία που παίρνει από το κάθε αντικείμενο μπορεί να προτείνει πολύ συγκεκριμένα αντικείμενα στους χρήστες που ταιριάζουν στο προφίλ, ίδια ή παρόμοια με αντικείμενα που έχουν ήδη επιλέξει. Έτσι δεν βοηθούν το χρήστη να ανακαλύψει καινούρια αντικείμενα, διαφορετικά από αυτά που έχει ήδη επιλέξει αλλά μπορεί να τον ενδιαφέρουν στον ίδιο βαθμό.

- Προσέγγιση με βάση τη συλλογικότητα

Στην τεχνική σύστασης με βάση τη συλλογικότητα, η ανάλυση εστιάζει στους χρήστες. Δηλαδή, τώρα, το προφίλ του κάθε χρήστη σχηματίζεται από τις βαθμολογίες που έχει δώσει σε κάθε αντικείμενο. Στη συνέχεια, γίνεται συσχέτιση των χρηστών σε σχέση με τον παρόμοιο τρόπο βαθμολογίας των ίδιων αντικειμένων. Το σύστημα αφού συσχετίσει τους χρήστες ανάλογα με τις βαθμολογίες τους στα ίδια αντικείμενα, προτείνει σε ένα χρήστη, αντικείμενα που δεν έχει αλλά τα έχουν βαθμολογήσει άλλοι χρήστες που ανήκουν στην ίδια ομάδα χρηστών.

Η προσέγγιση αυτή ξεπερνά κάποια από τα προβλήματα που έχει η μέθοδος που περιεγράφηκαν στην προηγούμενη παράγραφο. Για παράδειγμα, στην περίπτωση που ένα αντικείμενο δεν έχει επαρκείς πληροφορίες, με την προσέγγιση αυτή, μπορεί να προταθεί καθώς έχουν δώσει βαθμολογία οι άλλοι χρήστες. Ένα άλλο πλεονέκτημα αυτής της προσέγγισης είναι η ποιότητα που θα έχει το αντικείμενο. Δηλαδή ενώ στην προηγούμενη προσέγγιση, η σύσταση γίνεται μόνο με το περιεχόμενο, οπότε στην περίπτωση που έχουμε δύο αντικείμενα, το ένα καλής ποιότητας και το άλλο κακής, αλλά με το ίδιο περιεχόμενο, θα προταθούν και τα δύο. Αντίθετα στην παρούσα προσέγγιση τα αντικείμενα που προτείνονται έχουν βαθμολογηθεί από τους άλλους χρήστες, κάτι που μας δίνει και τον έλεγχο της ποιότητας του περιεχομένου του κάθε αντικειμένου. Τέλος, με την προσέγγιση αυτή, οι προτάσεις στους χρήστες μπορούν να αποτελούνται από διαφορετικά μεταξύ τους αντικείμενα, όσο αφορά στο περιεχόμενό τους, που όμως να ενδιαφέρουν το ίδιο τους χρήστες.

Η προσέγγιση με βάση τη συλλογικότητα, χωρίζεται σε δύο γενικές μεθοδολογίες, τις μεθόδους με βάση τη γειτονιά και τις μεθόδους που χρησιμοποιούν μοντέλα πρόβλεψης. Οι μέθοδοι που στηρίζονται στη γειτονιά αποθηκεύουν την πληροφορία χρήστης – βαθμολογία αντικειμένου και μπορούν να έχουν δύο διαφορετικές προσεγγίσεις, η πρώτη έχει ως κύριο αντικείμενο ανάλυσης τον χρήστη, ενώ η δεύτερη το αντικείμενο. Αυτές που στηρίζονται στους χρήστες, αξιολογούν το ενδιαφέρον κάποιου χρήστη χρησιμοποιώντας τις βαθμολογίες των άλλων χρηστών για το συγκεκριμένο αντικείμενο. Εκείνες που στηρίζονται στα αντικείμενα, προσπαθούν να προβλέψουν το ενδιαφέρον κάποιου χρήστη, ανάλογα τις βαθμολογίες που έχει δώσει σε παρόμοια αντικείμενα. Τα αντικείμενα τα οποία έχουν βαθμολογηθεί από διαφορετικούς χρήστες με τον ίδιο τρόπο θεωρούνται ίδια.

Αντίθετα από τα συστήματα γειτονιάς τα οποία χρησιμοποιούν αποθηκευμένες πληροφορίες απευθείας μέσα στη διαδικασία πρόβλεψης, τα συστήματα χρήσης μοντέλων χρησιμοποιούν αυτές της πληροφορίες για να εκπαιδεύσουν το μοντέλο που θα χρησιμοποιήσουν για την πρόβλεψη.

- Τα πλεονεκτήματα της προσέγγισης με βάση τη γειτονιά

Παρ’ όλο που πρόσφατες μελέτες έχουν δείξει ότι οι προσεγγίσεις που βασίζονται σε μοντέλα είναι ανώτερες από αυτές που βασίζονται στη γειτονιά σε σχέση με την ακρίβεια στην πρόβλεψη της βαθμολογίας, θα πρέπει να εξετάσουμε τη γενικότερη εμπειρία που προσφέρει η κάθε μία μέθοδος.

Οι προσεγγίσεις που βασίζονται σε μοντέλα διακρίνονται για το πόσο καλά μπορούν να χαρακτηρίζουν τις προτιμήσεις του χρήστη χωρίς όμως να τις κατηγοριοποιούν αυστηρά. Για παράδειγμα, σε ένα σύστημα προτάσεων βιβλίων το οποίο χρησιμοποιεί τις μεθόδους αυτές, μπορεί να ορίσει τους όρους «αστείο» και «ρομαντικό». Το σύστημα θα μπορεί να κάνει πολύ ακριβείς προβλέψεις για βιβλία «ρομαντικές κωμωδίες», όμως δεν θα καταφέρει να προτείνει στο χρήστη βιβλία που δεν κατατάσσονται σε αυτό το είδος όπως για παράδειγμα «μαύρη κωμωδία». Αυτού του είδους δυσκολίες μπορούν να ξεπεραστούν με την προσέγγιση που βασίζεται στη γειτονιά. Η προσέγγιση με βάση τη γειτονιά, μπορεί να διακρίνει συσχετίσεις μεταξύ των δεδομένων βάση της γειτνίασης. Έτσι μπορεί να προτείνει βιβλία πολύ διαφορετικά από αυτά που συνηθίζει ο χρήστης να βαθμολογεί ή και βιβλία που δεν γνωρίζει, αν κάποιος γείτονάς του έχει δώσει υψηλή βαθμολογία. Οι προτάσεις αυτές μπορεί να μην έχουν απόλυτη επιτυχία όμως βελτιώνουν την εμπειρία του χρήστη καθώς των βοηθούν να ανακαλύψει καινούρια βιβλία που πιθανώς να τον ενδιαφέρουν. Αναλυτικά, τα πλεονεκτήματα των μεθόδων που βασίζονται στη γειτονιά περιγράφονται παρακάτω.

a. Οι μέθοδοι αυτές είναι αρκετά απλές στην υλοποίηση τους.
b. Η πρόταση μπορεί να εξηγηθεί στο χρήστη. Δηλαδή, το σύστημα να του παρουσιάσει τις επιλογές και τις βαθμολογίες των γειτόνων του και πως προέκυψε η συγκεκριμένη πρόταση.
c. Ένα από τα πολύ σημαντικά πλεονεκτήματα των μεθόδων αυτών είναι η αποδοτικότητα. Δηλαδή, το σύστημα δεν χρειάζεται φάσεις εκπαίδευσης του αλγορίθμου οι οποίες συνήθως απαιτούν αρκετούς πόρους του συστήματος και πρέπει να γίνονται ανά τακτά χρονικά διαστήματα.
d. Η σταθερότητα είναι το επόμενο πιο σημαντικό πλεονέκτημα των μεθόδων αυτών. Δηλαδή, οι μέθοδοι αυτές δεν επηρεάζονται από την προσθήκη νέων χρηστών, νέων προϊόντων ή καινούριων βαθμολογιών.

- Μεθοδολογία για τη βάση Goodreads

Για τις ανάγκες της παρούσας διπλωματικής εργασίας, χρησιμοποιήθηκε η βάση δεδομένων της ηλεκτρονικής βιβλιοθήκης Goodreads (https://www.goodreads.com/). Η βιβλιοθήκη αυτή, φιλοξενεί μια ενημερωμένη και μεγάλη βάση δεδομένων βιβλίων, όλων των ειδών παραδείγματος χάριν λογοτεχνικά, επιστημονικά, πολιτικά κ.α. Οι χρήστες μπορούν να εγγραφθούν, να επιλέξουν τα βιβλία που έχουν διαβάσει και να τα προσθέσουν σε λίστες, να βαθμολογήσουν τα βιβλία αυτά με διακριτή βαθμολογία (1-5), να γράψουν την κριτική τους, καθώς και να φτιάξουν άλλες λίστες με τα βιβλία που θέλουν να διαβάσουν στο μέλλον. Τέτοιου είδους βάσεις δεδομένων έχουν εμφανιστεί σε

διάφορα είδη προϊόντων – τεχνών, όπως το Imdb (https://www.imdb.com/) και το Discogs (https://www.discogs.com/), τεράστιες βάσεις δεδομένων και ηλεκτρονικά καταστήματα ταινιών και μουσικής αντίστοιχα. Οι παραπάνω ηλεκτρονικές βιβλιοθήκες έχουν μεγάλη απήχηση στους χρήστες και κάθε μία απαριθμεί μερικά εκατομμύρια εγγεγραμμένων χρηστών.

Η επιτυχία και η απήχηση στους χρήστες των ηλεκτρονικών βιβλιοθηκών αυτών, από τη μία πλευρά οφείλεται στον αριθμό των μοναδικών προϊόντων που φιλοξενεί. Από την άλλη πλευρά, είναι η ευκολία χρήσης και αναζήτησης της πληροφορίας μέσα σ' αυτές τις αχανείς βάσεις πληροφοριών. Στο δεύτερο σημείο περιλαμβάνεται και το σύστημα σύστασης που ακολουθεί κάθε μία ηλεκτρονική βιβλιοθήκη ώστε να κάνει την εμπειρία του χρήστη ακόμα καλύτερη και την αναζήτηση της πληροφορίας πιο εύκολη.

Η ηλεκτρονική βιβλιοθήκη που επιλέξαμε να είναι το αντικείμενο της έρευνας αυτής, επιτρέπει στους χρήστες να γίνουν φίλοι μεταξύ τους, μια δυνατότητα που δυστυχώς δεν την επιτρέπουν οι υπόλοιπες ηλεκτρονικές βιβλιοθήκες. Για το λόγο αυτό θα προσπαθήσουμε να εφαρμόσουμε δύο μεθοδολογίες πάνω στα δεδομένα της βάσης αυτής χρησιμοποιώντας τη επιπλέον δυνατότητα που προσφέρει. Η πρώτη είναι η κλασική μέθοδος σύστασης με βάση τη γειτονιά, δηλαδή θα εφαρμόσουμε τον αλγόριθμο πρόβλεψης πιθανής βαθμολογίας σε κάποιο αντικείμενο με βάση τη γειτονιά του κάθε χρήστη η οποία διαμορφώνεται από το γράφημα που προκύπτει με τη σύνδεση τους με τη σχέση της φιλίας. Όπως παρουσιάστηκε και πιο πάνω, η μέθοδος αυτή δεν λαμβάνει υπόψη της τα αντικείμενα που έχει βαθμολογήσει ο χρήστης παρά μόνο τις σχέσεις μεταξύ τους όσο αφορά τη γειτονιά. Έτσι τα προτεινόμενα αντικείμενα θα είναι αντικείμενα που έχουν οι φίλοι τους στις λίστες τους, χωρίς να λαμβάνεται καθόλου υπόψη το είδος του αντικειμένου (πχ ρομαντική λογοτεχνία, φυσική επιστήμη κτλ). Για να εξετάσουμε αν η συστάσεις που θα κάνουμε θα είναι πιο ακριβείς, θα αξιολογήσουμε αρχικά τον τρόπο βαθμολόγησης του κάθε χρήστη σε σχέση με όλους τους άλλους. Δηλαδή για κάθε μοναδικό ζευγάρι χρηστών θα υπολογισθεί ένας βαθμός συσχέτισης με βάση τις βαθμολογίες που έχουν δώσει στα ίδια βιβλία – αντικείμενα. Στη συνέχεια στον αλγόριθμο που εφαρμόσαμε στη γειτονιά, θα προσθέσουμε και την παράμετρο του βαθμού συσχέτισης των χρηστών. Με αυτή τη μεθοδολογία θέλουμε να εξετάσουμε αν οι συνδυαστικές μέθοδοι θα δώσουν πιο ακριβή και πιο ενδιαφέροντα αποτελέσματα στον χρήστη.

- Εκτίμηση μιας «άγνωστης» βαθμολογίας του χρήστη (u) για το αντικείμενο (i)

Η εκτίμηση μιας πιθανής βαθμολογίας του χρήστη u για το αντικείμενο i, προκύπτει από την πρόσθεση της μέσης τιμής των βαθμολογιών του χρήστη u με το αποτέλεσμα της διαίρεσης του αθροίσματος του πηλίκου του βάρους μεταξύ του χρήστη u και του χρήστη v επί τη διαφορά της βαθμολογίας του χρήστη v στο αντικείμενο i μείον τη μέση τιμή των βαθμολογιών του χρήστη v, διά το άθροισμα της απόλυτης τιμής του βάρους μεταξύ του χρήστη u και του χρήστη v.

$$R_{ui} = \overline{r_u} + \frac{\sum_{V \in N(u|i)} W_{uv} * [R_{vi} - \overline{r_v}]}{\sum_{V \in N(u|i)} |W_{uv}|}$$

Εξίσωση 1: Εκτίμηση άγνωστης βαθμολογίας χρήστη

- Wφιλίας

Το βάρος που ορίζουμε όσο αφορά στη σύνδεση των χρηστών με τη σχέση φιλίας έχει δύο τιμές, 0 για δύο χρήστες που δεν είναι φίλοι και 1 για δύο χρήστες που είναι φίλοι.

- Wσυσχέτισης

Ο δεύτερος τύπος βάρους που χρησιμοποιούμε είναι ο βαθμός συσχέτισης μεταξύ των δύο χρηστών ο οποίος προκύπτει από τη διαίρεση του αθροίσματος του πηλίκου των βαθμολογιών των δύο χρηστών για κάθε κοινό βιβλίο, με το πηλίκο που προκύπτει από τον πολλαπλασιασμό της απόλυτης τιμής του αθροίσματος του πρώτου χρήστη με το άθροισμα της απόλυτης τιμής των βαθμολογιών του δεύτερου χρήστη.

$$W_{uv}^{c} = corr(\vec{Ru}, \vec{Rv}) = cos(\vec{Ru}, \vec{Rv})$$

$$= \frac{\vec{Ru} * \vec{Rv}}{\|\vec{Ru}\| * \|\vec{Rv}\|}$$

$$= \frac{\sum_{i=1}^{n} Ru(i) * Rv(i)}{\sqrt{\sum_{i=1}^{n} Ru(i)^2} * \sqrt{\sum_{i=1}^{n} Rv(i)^2}}$$

$$= \frac{\sum_{i \in I(u) \bigcap I(v)}^{n} Ru(i) * Rv(i)}{\sqrt{\sum_{i \in I(u) \bigcap I(v)}^{n} Ru(i)^2} * \sqrt{\sum_{i \in I(u) \bigcap I(v)}^{n} Rv(i)^2}}$$

Εξίσωση 2: Βάρος συσχέτισης χρηστών

2. Βάση Δεδομένων

- Συλλογή (Data Collation)

Για τη συλλογή των δεδομένων κατασκευάστηκαν εργαλεία τα οποία αναζητούν τα δεδομένα που χρειαζόμαστε στον Html κώδικα της ιστοσελίδας και αφού τα συγκεντρώσουν τα αποθηκεύουν σε αρχεία σε τοπικό φάκελο του ηλεκτρονικού υπολογιστή.

Η μέθοδος αυτή ονομάζεται web scraping. Το λογισμικό μπορεί να αποκτήσει πρόσβαση σε μία ιστοσελίδα άμεσα μέσω του πρωτοκόλλου HTTP ή μέσω ενός φυλλομετρητή, κατεβάζει τη σελίδα που κάλεσε και εξάγει τα δεδομένα. Συγκεκριμένα, κατασκευάστηκαν τρία προγράμματα (scripts) για τη συλλογή των χρηστών, τη συλλογή των φίλων των χρηστών και τη συλλογή της λίστας των βιβλίων.

- Προετοιμασία (Data Preparation)

Αφού έχουμε εξάγει τα δεδομένα που μας ενδιαφέρουν από την ιστοσελίδα, ενώνουμε τα δεδομένα που διατηρούνται σε πίνακα κειμένου, σε ένα μεγάλο κείμενο το οποίο είναι στη μορφή κειμένου csv με τα δεδομένα χωρισμένα με κόμμα (,). Στη συνέχεια αποθηκεύουμε τη μεταβλητή αυτή σε αρχείο σε τοπικό φάκελο του ηλ. Υπολογιστή.

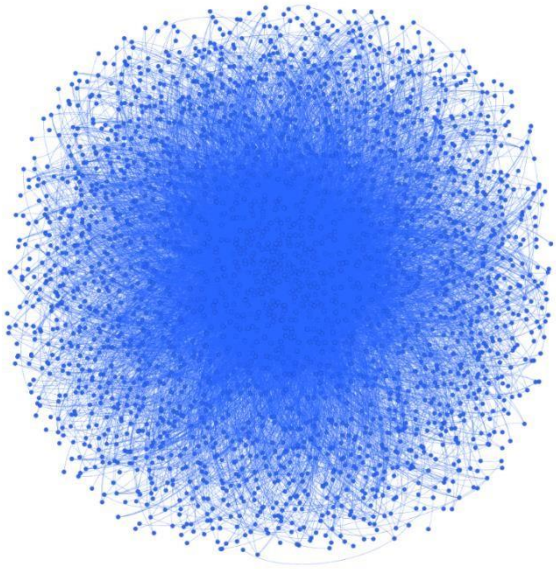
- Αποθήκευση (Data Storage)

Για την αποθήκευση των δεδομένων θα κατασκευάσουμε μια βάση δεδομένων. Τα βασικά δεδομένα που θα πρέπει να διατηρήσουμε είναι τα βιβλία και οι χρήστες. Έτσι προκύπτουν δύο πίνακες, αυτός των βιβλίων ο οποίος περιέχει ένα χαρακτηριστικό αριθμό ο οποίος είναι το πρωτεύον κλειδί του πίνακα και ένα πεδίο το οποίο κρατάει το όνομα του βιβλίο. Στον πίνακα των χρηστών αποθηκεύουμε δύο βασικές πληροφορίες τη διεύθυνση του χρήστη η οποία θα είναι και το πρωτεύον κλειδί, το όνομα του χρήστη. Κατά την επεξεργασία των δεδομένων, θα χρειαστούμε δύο ακόμα πληροφορίες τις οποίες μπορούμε να τις υπολογίσουμε κατά τη διάρκεια που εκτελούνται οι αλγόριθμοι. Επειδή δεν έχουμε ροή καινούριων δεδομένων και για να εκτελεστούν γρηγορότερα οι αλγόριθμοι, θα αποθηκεύσουμε των αριθμό των βιβλίων που έχει βαθμολογήσει ο χρήστης και το μέσο όρο της βαθμολογίας που έχει δώσει.

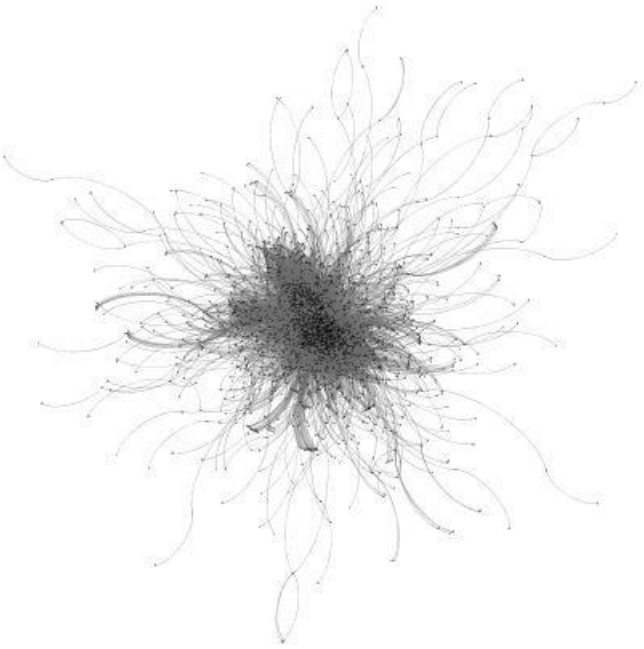
Πέρα από τους βασικούς πίνακες, θα χρειαστούμε και άλλους δύο πίνακες οι οποίοι ουσιαστικά μας περιγράφουν τη σχέση μεταξύ των δύο προαναφερθέντων πινάκων. Ο πρώτος πίνακας συνδέει τους χρήστες με τα βιβλία. Περιέχει το user id από των πίνακα των χρηστών, book id από τον πίνακα των βιβλίων. Τα δύο αυτά πεδία, αποτελούν τα ξένα κλειδιά από τους πίνακες users και books αντιστοίχως και μαζί αποτελούν το σύνθετο κλειδί του πίνακα. Τέλος, ο πίνακας αυτός κρατάει και τη βαθμολογία του χρήστη για το βιβλίο. Ο τελευταίος πίνακας που θα δημιουργήσουμε είναι ο πίνακας που συνδέει τους χρήστες με τους φίλους τους. Περιλαμβάνει δύο πεδία, το user id και το friend id, που αποτελούν ξένα κλειδιά από τον πίνακα users και μαζί είναι το σύνθετο κλειδί για τον πίνακα. Και στον πίνακα αυτό για λόγους ταχύτητας εκτέλεσης των αλγορίθμων θα δημιουργήσουμε ακόμα ένα πεδίο το οποίο θα αποθηκεύει το βάρος μεταξύ των δύο χρηστών.

- Δίκτυο (Network)

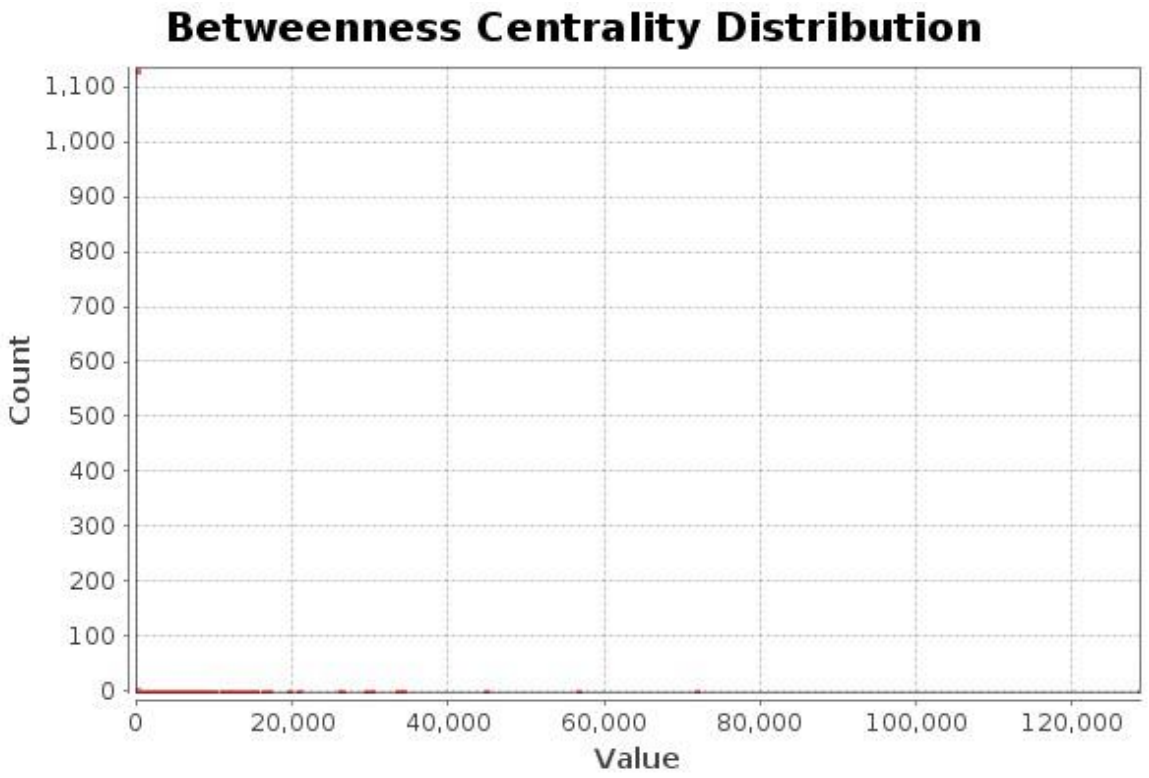
Το δίκτυο που έχουμε δημιουργήσει είναι ένα μεγάλο υποδίκτυο του δικτύου της βάσης του Goodreads και αποτελείται από 3492 εγγεγραμμένους χρήστες και 14784 σχέσεις φιλίας μεταξύ των χρηστών. Τα βιβλία που πήραμε είναι 316288, ενώ τα βιβλία στις λίστες των χρηστών είναι συνολικά 971560. Στην εικόνα φαίνεται η γραφική αναπαράσταση των χρηστών και των συνδέσμων μεταξύ τους με βάση τη σχέση της φιλίας.



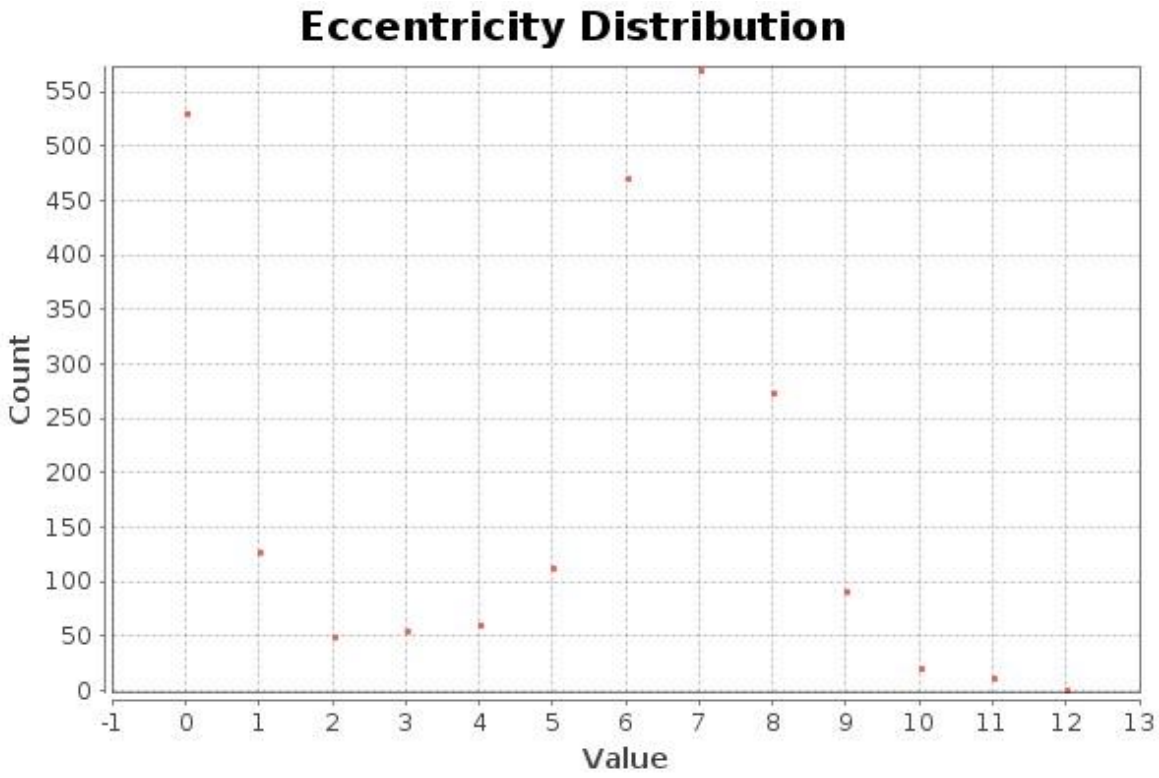
Εικόνα 1: Noverlap Algorithm



Εικόνα 2: Force Atlas 2 Algorithm



Γράφημα 1: Betweenness Centrality Distribution



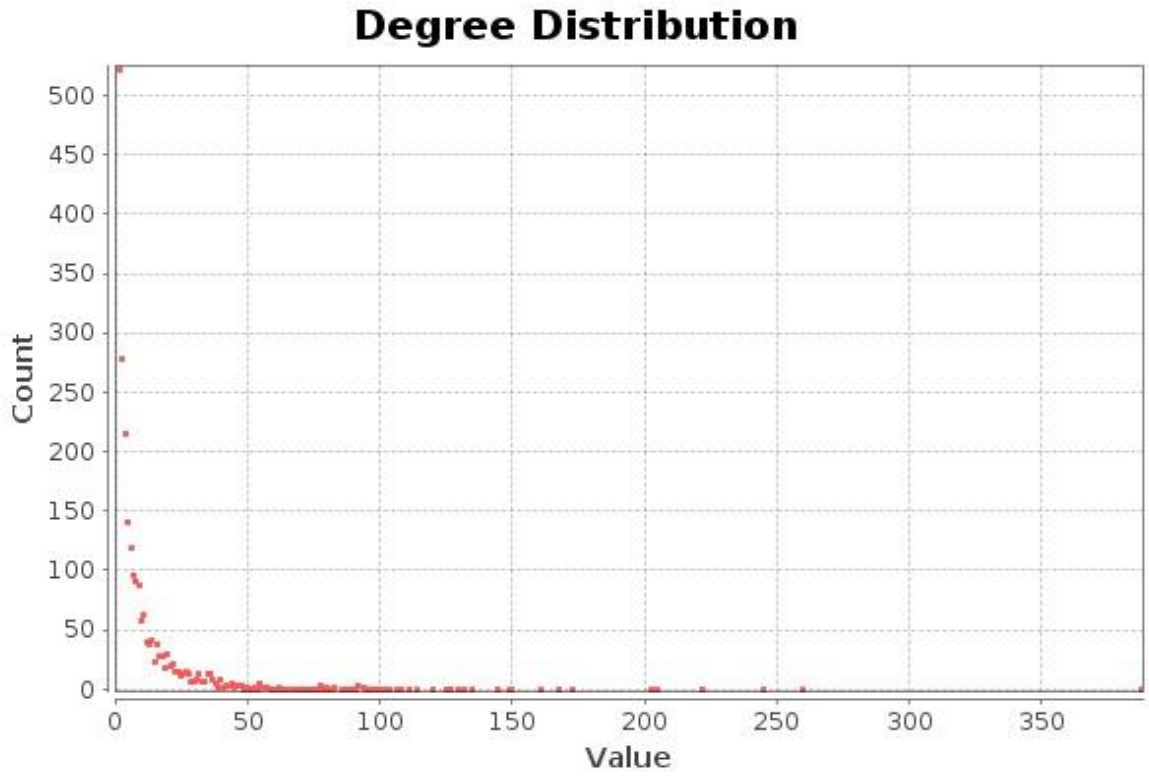
Γράφημα 2: Eccentricity Distribution

Μεγαλύτερο γεωδεσικό μονοπάτι: 12

Μέσο μήκος μονοπατιού: 3.311204205703008

Betweenness Centrality είναι η μέτρηση η οποία υπολογίζει τη συχνότητα με την οποία ένας κόμβος εμφανίζεται στα συντομότερα μονοπάτια μεταξύ των κόμβων ενός δικτύου.

Eccentricity είναι η μέτρηση η οποία υπολογίζει την απόσταση ενός κόμβου από τον πιο μακρινό του κόμβο. Στο διάγραμμα Eccentricity Distribution απεικονίζεται αυτή η μέτρηση για όλους τους κόμβους του δικτύου.



Γράφημα 3: Degree Distribution

Ο βαθμός ενός κόμβου είναι ο αριθμός των ακμών που έχει, ανεξάρτητα την κατεύθυνσή τους, δηλαδή αν πηγαίνουν από άλλο κόμβο προς αυτόν που εξετάζουμε ή φεύγουν από αυτόν με κατεύθυνση κάποιον άλλο κόμβο. Στην περίπτωση της παρούσας βάσης δεδομένων οι ακμές συμβολίζουν τη σύνδεση μεταξύ των χρηστών με το δεσμό της φιλίας. Η σχέση αυτή είναι αμφίδρομη, δηλαδή δεν έχει κατεύθυνση.

3. Πειραματικά αποτελέσματα

Για την άντληση των πειραματικών αποτελεσμάτων θεωρήσαμε σαν άγνωστη τη βαθμολογία που έχει δώσει ο κάθε χρήστης για το καθένα βιβλίο. Έτσι για κάθε βιβλίο κάθε χρήστη εφαρμόσαμε τους δύο αλγορίθμους όπως θα κάναμε δηλαδή σε μια πραγματική περίπτωση που θα αναζητούσαμε μια πιθανή βαθμολογία για κάποιο αντικείμενο που δεν έχει βαθμολογηθεί. Έτσι για

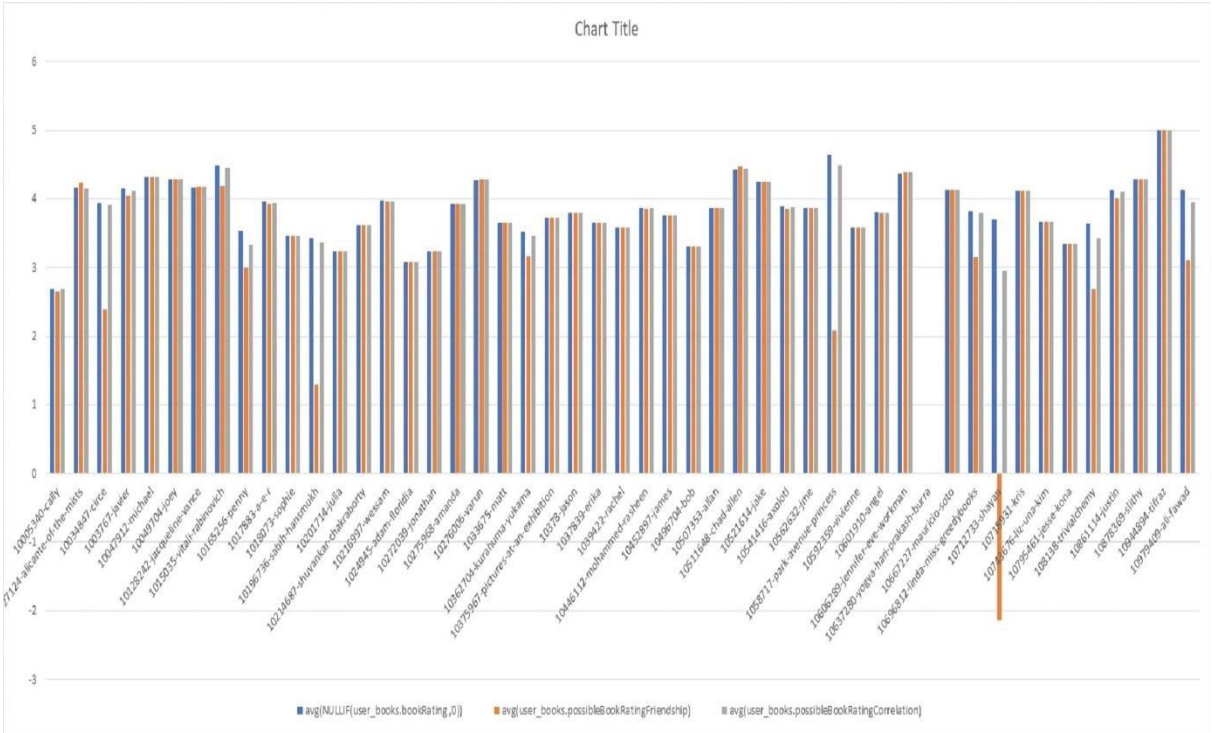
κάθε βιβλίο έχουμε την πραγματική βαθμολογία που έδωσε ο χρήστης και τις δύο πιθανές βαθμολογίες, μία από κάθε αλγόριθμο. Απεικονίζοντας τα δεδομένα αυτά σε γραφήματα, όπως θα δούμε παρακάτω, μπορούμε να αντλήσουμε συμπεράσματα σχετικά με την ακρίβεια και την αποτελεσματικότητα του κάθε αλγορίθμου. Στη συνέχεια θα υπολογίσουμε για κάθε χρήστη το μέσο απόλυτο σφάλμα, το οποίο θα μας δώσει περισσότερα στοιχεία για την αποτελεσματικότητα των αλγορίθμων.

userID	bookID	bookRating	possibleBookRatingFriendship	possibleBookRatingCorrelation
10027124-alicante-ofthe-mists	10	4	5.87	4.32
10027124-alicante-ofthe-mists	88	4	4.15	4.15
10027124-alicante-ofthe-mists	109	4	1.55	4.13
10027124-alicante-ofthe-mists	122	4	4.15	4.15
10027124-alicante-ofthe-mists	127	4	5.87	4.32
10027124-alicante-ofthe-mists	185	5	2.87	4.02
10027124-alicante-ofthe-mists	187	4	3.87	4.12
10027124-alicante-ofthe-mists	190	4	4.15	4.15
10027124-alicante-ofthe-mists	283	4	4.15	4.15
10027124-alicante-ofthe-mists	303	4	0.55	3.81
10027124-alicante-ofthe-mists	336	5	4.15	4.15
10027124-alicante-ofthe-mists	338	4	2.55	3.94
10027124alicante-ofthe-mists	436	3	4.15	4.15

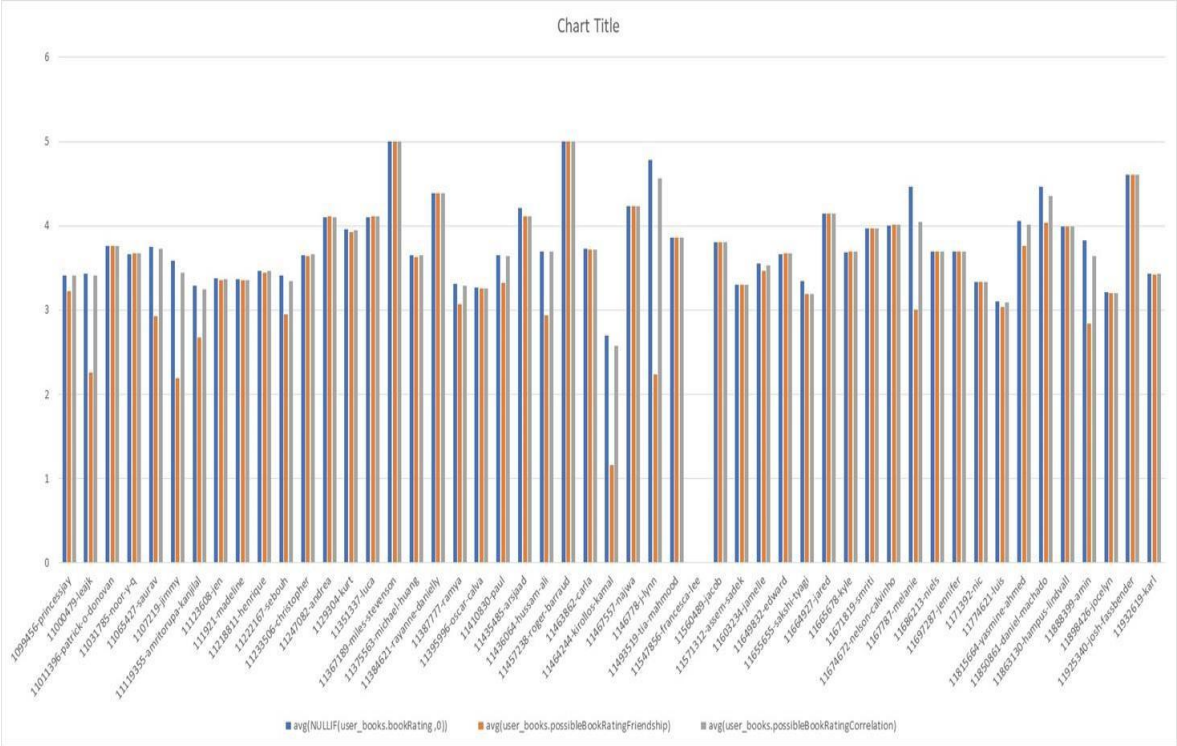
Στον παραπάνω πίνακα βλέπουμε τα δεκατρία πρώτα βιβλία που υπάρχουν στη λίστα του χρήστη «10027124-alicante-of-the-mists» (τυχαίο παράδειγμα).

userID	Μέσος όρος βαθμολογίας χρηστών	Μέσος όρος πιθανής βαθμολογίας με τον αλγόριθμο της γειτονιάς	Μέσος όρος πιθανής βαθμολογίας με τον αλγόριθμο του βαθμού συσχέτισης
10005340-cally	2.6935	2.646024	2.687786
10027124-alicanteof-the-mists	4.1536	4.235143	4.152679
10034847-circe	3.9412	2.389947	3.904849
1003767-javier	4.1402	4.044832	4.115906
10047912-michael	4.3158	4.32	4.32
10049704-joey	4.2852	4.29	4.29
10128242jacqueline-vance	4.1669	4.17	4.17
1015035-vitalirabinovich	4.4762	4.189137	4.456269
10165256-penny	3.5275	2.996339	3.339746
1017883-a-e-r	3.9595	3.923909	3.934482

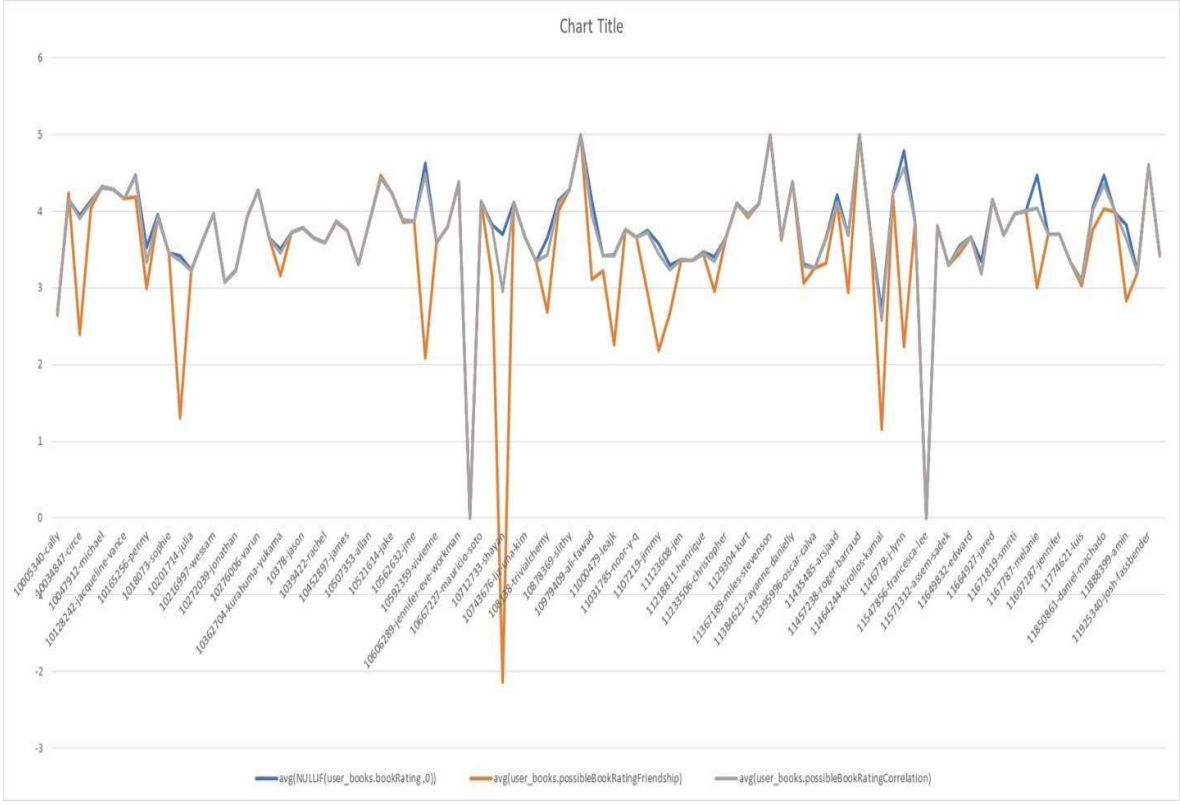
Στον παραπάνω πίνακα φαίνεται ο μέσος όρος της πραγματικής βαθμολογίας των πρώτων δέκα χρηστών της βάσης, τον μέσο όρο της πιθανής βαθμολογίας των χρηστών που προέκυψαν από τον αλγόριθμο της γειτονιάς και τέλος τον μέσο όρο της πιθανής βαθμολογίας των χρηστών που προέκυψαν από τον συνδυαστικό αλγόριθμο της γειτονιάς με το βαθμό συσχέτισης.



Πίνακας 1: Πρώτοι 50 χρήστες



Πίνακας 2: 50 - 100 χρήστες



Πίνακας 3: Πρώτοι 50 χρήστες

Στα παραπάνω γραφήματα απεικονίζεται σε τρεις στήλες ο μέσος όρος της πραγματικής βαθμολογίας (μπλε στήλη), ο μέσος όρος της πιθανής βαθμολογίας με τον αλγόριθμο της γειτονιάς (πορτοκαλί στήλη) και ο μέσος όρος της πιθανής βαθμολογίας με βάση τη γειτονιά και το βαθμό

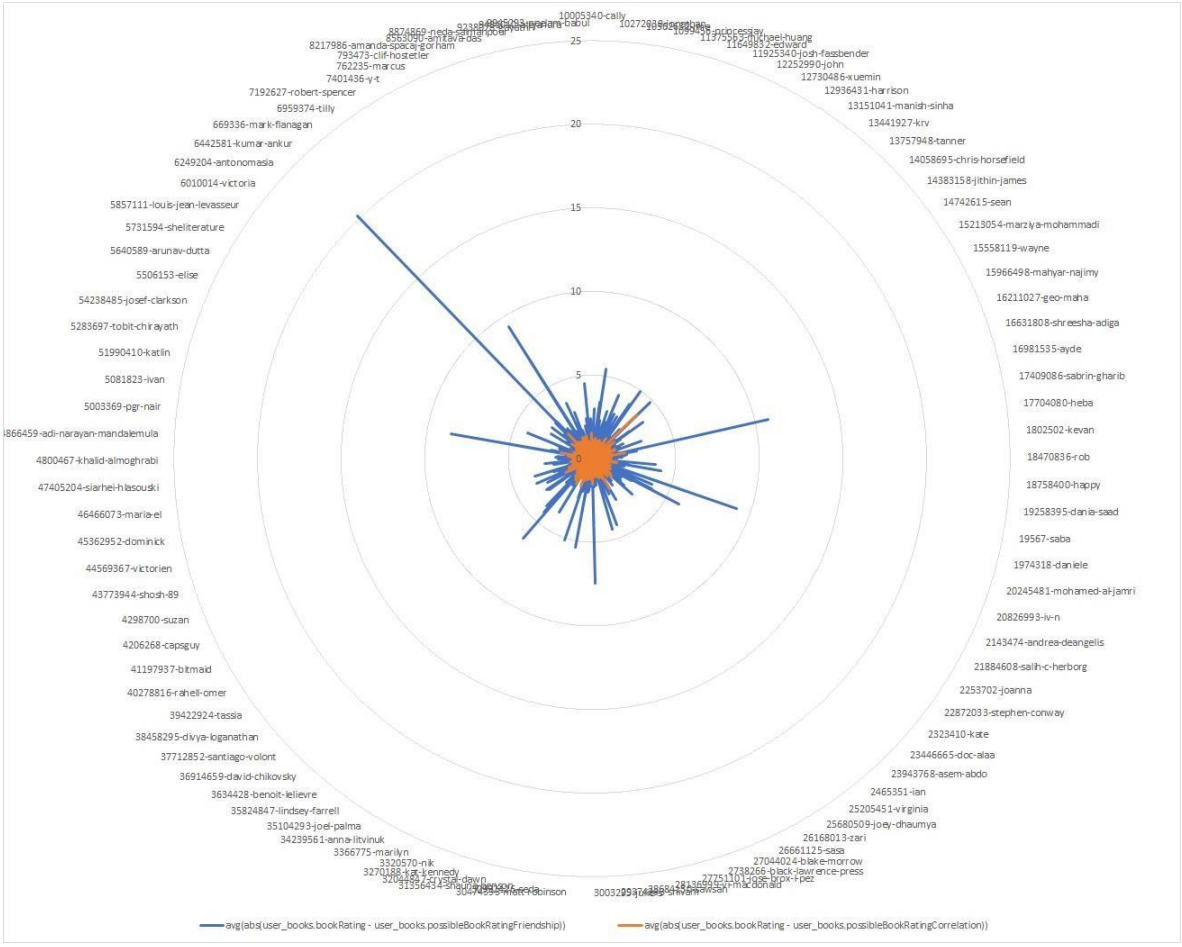
συσχέτισης (καφέ στήλη) για τους πρώτους εκατό χρήστες της βάσης (Πίνακας 1 1-50, πίνακας 2 51 – 100). Στον τρίτο πίνακα βλέπουμε την απεικόνιση των πρώτων πενήντα χρηστών σε γράφημα διακύμανσης τιμής (μπλε γραμμή – πραγματική βαθμολογία, πορτοκαλί γραμμή – πιθανή βαθμολογία γειτονιάς, καφέ γραμμή – πιθανή βαθμολογία συσχέτισης).

userID	Μέσο απόλυτο σφάλμα (γειτονιά)	Μέσο απόλυτο σφάλμα (βαθμός συσχέτισης)
10005340-cally	0.858276	0.799732
10027124-alicante-of-themists	0.844082	0.623071
10034847-circe	1.223529	0.812941
1003767-javier	0.818571	0.757593
10047912-michael	0.576842	0.576842
10049704-joey	0.687109	0.687109
10128242-jacqueline-vance	0.652483	0.652483
1015035-vitali-rabinovich	1.058358	0.718555
10165256-penny	1.434591	0.896745
1017883-a-e-r	0.544595	0.477838
1018073-sophie	0.6848	0.6848
10196736-sabih-hansmukh	2.987637	0.80716
10201714-julia	1.09487	1.09487
10214687-shuvankarchakraborty	0.686456	0.590759
10216997-wessam	0.616712	0.616712
1024945-adam-floridia	0.992807	0.992807
10272039-jonathan	0.824359	0.824359
10275968-amanda	0.68	0.68
10276006-varun	0.617	0.617
1033675-matt	0.986792	0.986792
10362704-kurahuma-yukama	1.045625	0.771518
10375967-pictures-at-anexhibition	0.621067	0.621067
10378-jason	0.758571	0.758571
1037839-erika	0.879827	0.879827
1039422-rachel	1.119796	1.119796
10446112-mohammedrasheen	0.5952	0.560867
10452897-james	0.65	0.65
10496704-bob	0.533077	0.533077
10507353-allan	0.631341	0.631341
10511648-chad-allen	0.858571	0.617143
10521614-jake	0.648571	0.648571
10541416-axolotl	0.834933	0.744367

10562632-jme	0.510779	0.510779
1058717-park-avenueprincess	3.399018	0.538379
10592359-vivienne	0.746782	0.746782
10601910-angel	0.632979	0.632979
10606289-jennifer-eveworkman	0.758985	0.758985
10667227-mauricio-soto	0.593684	0.593684
10696812-linda-missgreedybooks	1.269193	0.68721
10712733-shayan	5.442294	1.217056
10718931-kris	0.693277	0.693277
10743676-liz-una-kim	0.767949	0.767949
10795461-jesse-kona	0.721287	0.721287
108138-trivialchemy	1.555732	0.951098
10861114-justin	0.674659	0.591932
10878369-slithy	0.41	0.41
10944894-tifraz	0	0
10979409-ali-fawad	1.893692	0.776615
1099456-princessjay	0.822284	0.684537
11000479-leajk	1.489391	0.842912
11011396-patrick-o-donovan	0.563158	0.563158
11031785-noor-y-q	0.831667	0.831667
11065427-saurav	1.639863	0.616027
1107219-jimmy	2.32236	0.984037
11119355-amritorupa-kanjilal	1.530635	1.048824
11123608-jen	1.019373	0.990726
111921-madeline	0.922768	0.922768
11218811-henrique	0.855338	0.840456
11222167-sebouh	1.351313	0.799875
11233506-christopher	0.707587	0.670465
11247082-andrea	0.697667	0.591111
1129304-kurt	0.603165	0.57193
11351337-luca	0.559725	0.559725
11367189-miles-stevenson	0	0
11375563-michael-huang	0.824766	0.814439
11384621-rayanne-danielly	0.759709	0.759709
11387777-ramya	1.093276	0.89319

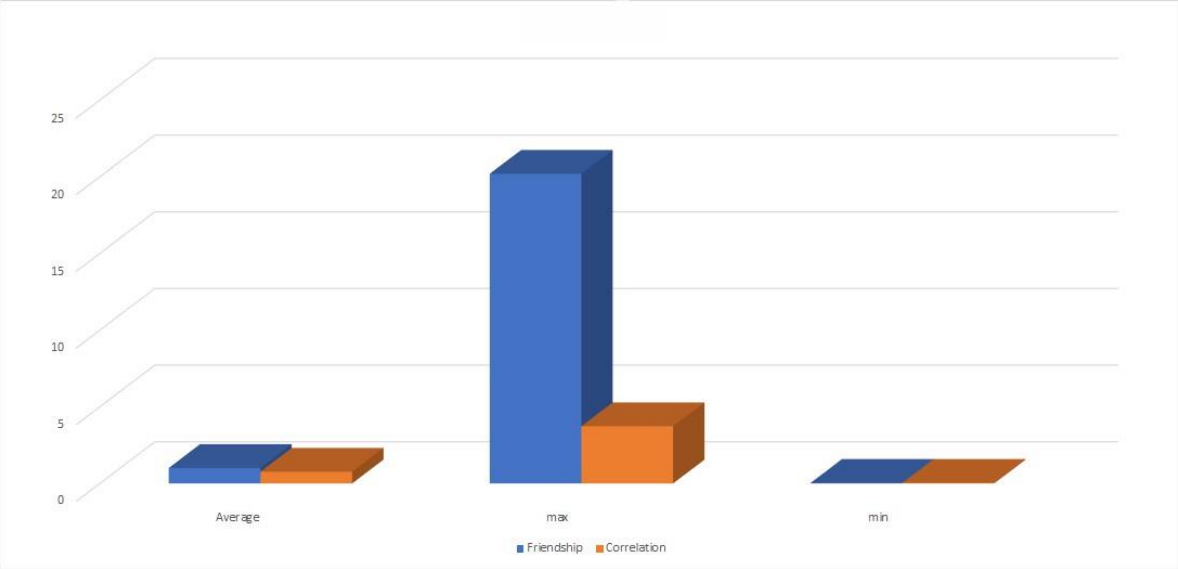
11395996-oscar-calva	0.986316	0.986316
11410830-paul	0.937624	0.627675

Ο παραπάνω πίνακας απεικονίζει τους εβδομήντα πρώτους χρήστες του συγκεντρωτικού πίνακα με το μέσο απόλυτο σφάλμα. Περιέχει την τιμή για το μέσο απόλυτο σφάλμα, στη δεύτερη στήλη βάση των πιθανών βαθμολογιών που προέκυψε από τον αλγόριθμο της γειτονιάς, ενώ στην τρίτη στήλη βάση των πιθανών βαθμολογιών που προέκυψε από τον αλγόριθμο του βαθμού συσχέτισης.



Γράφημα 4: Μέσο Απόλυτο Σφάλμα (κυκλικό)

Στον πίνακα 4 φαίνονται σε κυκλικό διάγραμμα οι τιμές του μέσου απόλυτου σφάλματος όλων των χρηστών του δικτύου. Παρατηρούμε ότι το αποτύπωμα του ΜΑΣ που προέκυψε από τον αλγόριθμο γειτονιάς – συσχέτισης, συγκεντρώνεται στο κέντρο του διαγράμματος, γύρο από το μηδέν, ενώ οι μεγαλύτερες τιμές είναι αρκετά χαμηλότερες του πέντε με ελάχιστες εξαιρέσεις (πορτοκαλί αποτύπωμα). Από την άλλη πλευρά, το αποτύπωμα του αλγορίθμου της γειτονιάς (μπλε αποτύπωμα) έχει μεγαλύτερες τιμές καθώς ξεπερνά σε έκταση το πορτοκαλί αποτύπωμα, ενώ οι τιμές είναι πιο κοντά στο πέντε και αρκετές το ξεπερνούν.

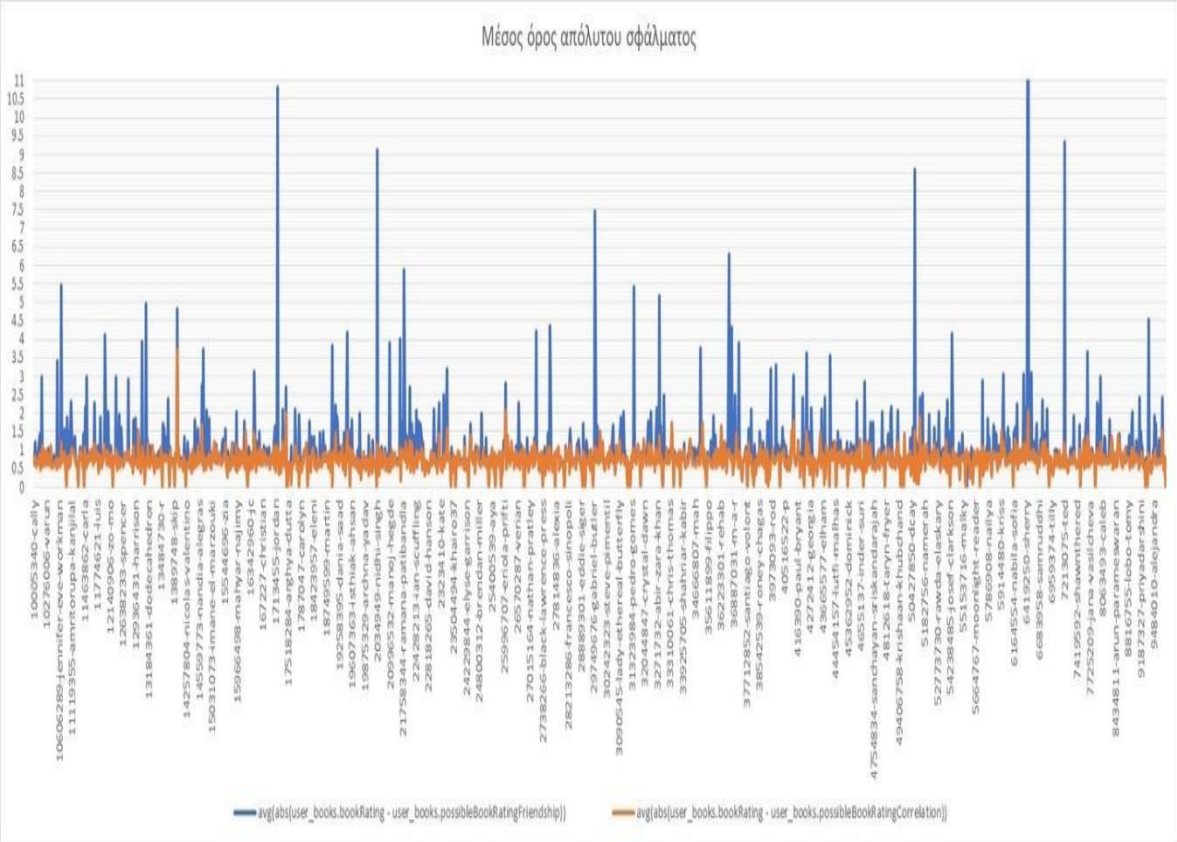


Γράφημα 5: Μέση, μέγιστη, ελάχιστη τιμή ΜΑΣ

Ο πίνακας 5 παρουσιάζει στατιστικά στοιχεία των τιμών του ΜΑΣ όλων των χρηστών. Στο πρώτο δίστηλο παρουσιάζεται οι μέσες τιμές του ΜΑΣ του δικτύου για όλους τους χρήστες και για τους δύο αλγορίθμους. Η μέση τιμή του ΜΑΣ του αλγορίθμου της γειτονιάς είναι 1.008115882 ενώ της γειτονιάς – συσχέτισης 0.764886008. Από τη μέση τιμή των αλγορίθμων φαίνεται ότι ο αλγόριθμος γειτονιάς – συσχέτισης έχει μεγαλύτερη ακρίβεια στην πρόβλεψη καθώς η τιμή είναι μικρότερη.

Στο δεύτερο δίστηλο, παρουσιάζεται η μέγιστη τιμή του ΜΑΣ, η οποία για τον πρώτο αλγόριθμο είναι 20.193542 ενώ για τον δεύτερο 3.73. Οι τόσο μεγάλες τιμές στο ΜΑΣ δείχνουν ότι ο πρώτος αλγόριθμος μπορεί να πέσει σε σφάλμα, εντελώς λανθασμένη εκτίμηση, κάτι που δεν φαίνεται να συμβαίνει στο δεύτερο.

Το τρίτο δίστηλο παρουσιάζει την ελάχιστη τιμή του ΜΑΣ και για τους δύο αλγόριθμους (εξαιρέσαμε τις μηδενικές τιμές καθώς οφείλονται σε ελλιπή στοιχεία). Οι ελάχιστες τιμές και για τους δύο αλγορίθμους είναι 0.04, μια τιμή πολύ κοντά στο μηδέν κάτι που φανερώνει και εδώ ελλιπή στοιχεία.



Γράφημα 6: ΜΑΣ για τους πρώτους 100 χρήστες (στήλες)

Στο παραπάνω πίνακα απεικονίζεται σε γράφημα στηλών, η τιμή του μέσου απόλυτου σφάλματος για τους πρώτους εκατό χρήστες της βάσης (μπλε στήλες – μέσο απόλυτο σφάλμα γειτονιάς, πορτοκαλί στήλες – μέσο απόλυτο σφάλμα συσχέτισης). Για το διάγραμμα χρησιμοποιήθηκαν οι εκατό πρώτοι χρήστες ώστε να είναι πιο ευανάγνωστο. Ακόμα και από αυτές τις τιμές μπορούμε να καταλάβουμε ότι οι τιμές του δεύτερου αλγορίθμου είναι πιο κοντά στο μηδέν οπότε το σφάλμα είναι μικρότερο.

4. Συμπεράσματα

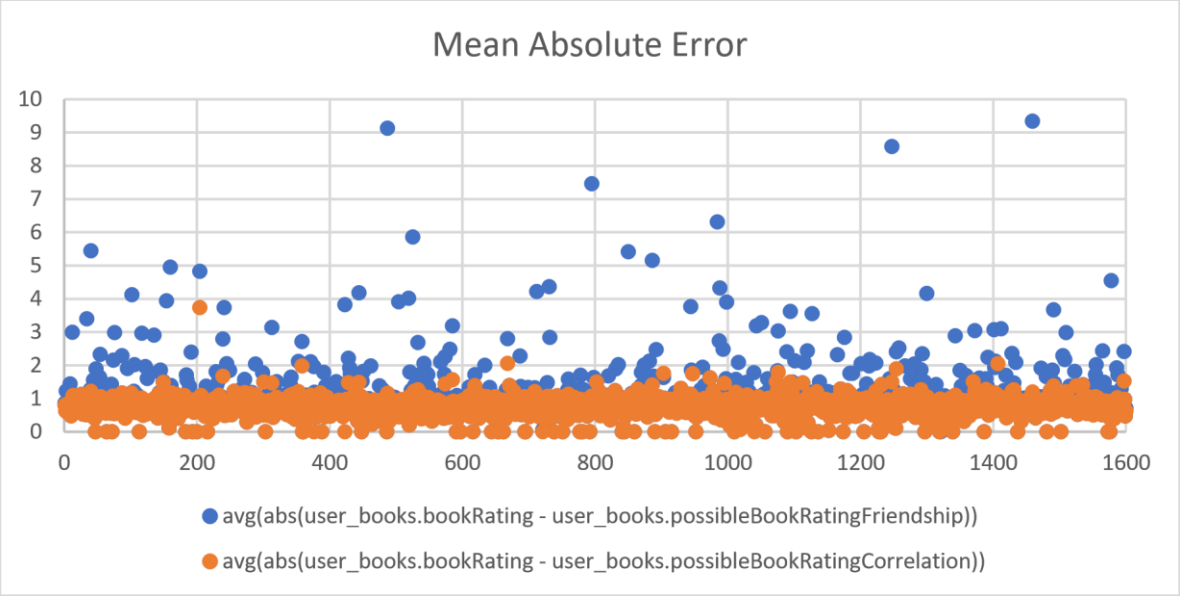
Στην προηγούμενη ενότητα έγινε περιγραφή του τρόπου δοκιμής των δύο αλγορίθμων που χρησιμοποιήθηκαν με τη βάση του Goodreads. Αφού λοιπόν έγινε η δοκιμή των αλγορίθμων, για κάθε χρήστη προκύπτουν δύο πιθανές βαθμολογίες, μία από κάθε αλγόριθμο, για κάθε βιβλίο που έχει βαθμολογήσει. Η πρώτη σύγκριση που εφαρμόσαμε είναι η σύγκριση στον μέσο όρο των βαθμολογιών (πραγματικών και πιθανών) του κάθε χρήστη. Όπως βλέπουμε στις τιμές και στο γράφημα, οι μέσοι όροι των βαθμολογιών του δεύτερου αλγορίθμου, σχεδόν ταυτίζονται με τους μέσους όρους των πραγματικών βαθμολογιών, οι διαφορές κυμαίνονται μεταξύ του 0 μέχρι 0,4 ενώ δεν παρατηρούνται εσφαλμένες τιμές. Από την άλλη πλευρά, οι τιμές των μέσων όρων των πιθανών βαθμολογιών που προέκυψαν με τη χρήση του πρώτου αλγορίθμου, παρουσιάζουν μεγαλύτερες διαφορές από τις τιμές των μέσων όρων των πραγματικών βαθμολογιών. Συγκεκριμένα, οι διαφορές κυμαίνονται από 0 μέχρι και 3,5 μονάδων, μια αρκετά μεγάλη διαφορά που δίνει λανθασμένη πρόβλεψη. Επίσης με τη χρήση του αλγορίθμου αυτού παρατηρήθηκαν και αρκετές λανθασμένες τιμές, καθώς προέκυψαν αρνητικές τιμές.

Η δεύτερη σύγκριση που εφαρμόστηκε είναι ο υπολογισμός του Μέσου απόλυτου σφάλματος.

$$MAE = \frac{\sum_{i=1}^{n} | y_i - x_i |}{n}$$

Εξίσωση 3: Mean Absolute Error

Το μέσο απόλυτο σφάλμα, είναι η τιμή που μας δείχνει τη διαφορά της πραγματικής τιμής από την πιθανή τιμή. Αναλυτικά, είναι το άθροισμα της απόλυτης τιμής της διαφοράς της πραγματικής από την πιθανή βαθμολογία για κάθε αντικείμενο, διά το πλήθος των αντικειμένων.



Γράφημα 7: ΜΑΣ για όλους τους χρήστες

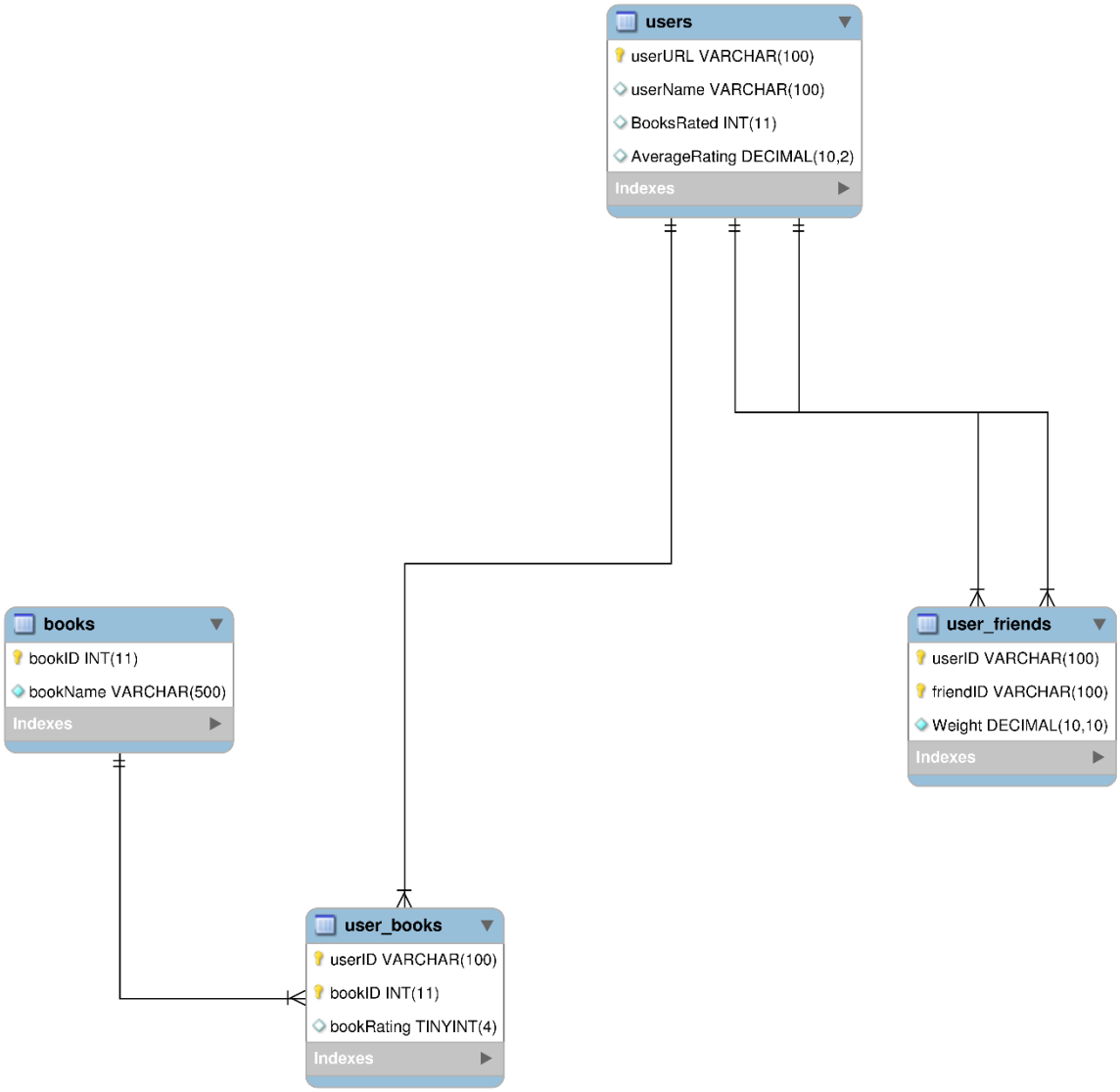
Στο παραπάνω γράφημα βλέπουμε μια απεικόνιση του μέσου απόλυτου σφάλματος, οι μπλε κουκίδες αναπαριστούν τις τιμές που προέκυψαν από τον πρώτο αλγόριθμο (γειτονιά) ενώ οι πορτοκαλί κουκίδες αναπαριστούν τις τιμές που προέκυψαν από το δεύτερο αλγόριθμο (γειτονιά – βαθμός συσχέτισης). Το αποδεκτό εύρος των τιμών για το Μ.Α.Σ. είναι από 0 μέχρι το μισό της μέγιστης τιμής που μπορεί να πάρει μια βαθμολογία, δηλαδή 5. Στο διάγραμμα λοιπόν παρατηρούμε ότι όλες σχεδόν οι τιμές βρίσκονται μέσα στα όρια αυτά και για τους δύο αλγορίθμους με ελάχιστες εξαιρέσεις κυρίως από τον πρώτο αλγόριθμο. Παρ' όλα αυτά, παρατηρούμε ότι οι τιμές του σφάλματος που μας δίνει ο δεύτερος αλγόριθμος είναι πιο κοντά στο 0, δηλαδή η τιμή του σφάλματος είναι μικρότερη, το οποίο συνεπάγεται ότι ο δεύτερος αλγόριθμος κάνει πιο ακριβή πρόβλεψη.

5. Επεκτασιμότητα

Ο αλγόριθμος που συνδυάζει τη γειτονιά με το βαθμό συσχέτισης, λαμβάνει υπόψη μόνο τους άμεσους γείτονες στην αξιολόγηση και πρόβλεψη της βαθμολογίας κάποιου αντικειμένου. Ο αλγόριθμος θα μπορούσε να εξελιχθεί λαμβάνοντας υπόψη μακρινότερους κόμβους, δηλαδή κόμβους που απέχουν από τον κόμβο που εξετάζουμε δύο και περισσότερα βήματα. Σε αυτή την περίπτωση πέρα από το βάρος που προκύπτει από το βαθμό συσχέτισης, θα πρέπει να εισάγουμε ένα νέο βάρος που θα συμβολίζει την απόσταση των δύο συσχετιζόμενων κόμβων, δηλαδή άλλο βάρος για τους κόμβους που απέχουν μία ακμή, άλλο για αυτούς με δύο και ούτω καθεξής.

6. Παράρτημα

- Σχεδιάγραμμα οντοτήτων – συσχετίσεων



Εικόνα 3: Σχεδιάγραμμα Οντοτήτων - Συσχετίσεων

- Κώδικας βάσης SQL

```
CREATE DATABASE  IF NOT EXISTS `goodreads`;

USE `goodreads`;


CREATE TABLE `books` (
`bookID` int(11) NOT NULL AUTO_INCREMENT,

`bookName` varchar(500) COLLATE utf8_unicode_ci NOT NULL,

PRIMARY KEY (`bookID`)

) ENGINE=InnoDB AUTO_INCREMENT=316291 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

CREATE TABLE `user_books` (
```

```
`userID` varchar(100) COLLATE utf8_unicode_ci NOT NULL,

`bookID` int(11) NOT NULL,

`bookRating` tinyint(4) DEFAULT NULL,

PRIMARY KEY (`userID`,`bookID`),

KEY `user_books_books_bookID_fk` (`bookID`),

CONSTRAINT `user_books_books_bookID_fk` FOREIGN KEY (`bookID`) REFERENCES `books` (`bookID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `user_books_users_userURL_fk` FOREIGN KEY (`userID`) REFERENCES `users` (`userURL`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `user_friends` (

`userID` varchar(100) COLLATE utf8_unicode_ci NOT NULL,

`friendID` varchar(100) COLLATE utf8_unicode_ci NOT NULL,

`Weight` decimal(10,10) NOT NULL DEFAULT '0.0000000000',

PRIMARY KEY (`userID`,`friendID`),

KEY `user_friends_friends_userURL_fk` (`friendID`),

CONSTRAINT `user_friends_friends_userURL_fk` FOREIGN KEY (`friendID`) REFERENCES `users` (`userURL`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `user_friends_users_userURL_fk` FOREIGN KEY (`userID`) REFERENCES `users` (`userURL`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `users` (

`userURL` varchar(100) COLLATE utf8_unicode_ci NOT NULL,

`userName` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,

`BooksRated` int(11) DEFAULT NULL,

`AverageRating` decimal(10,2) DEFAULT NULL,
PRIMARY KEY (`userURL`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- Κώδικας ανάκτησης χρηστών

```
# Εισαγωγή βιβλιοθήκης φυλλομετρητή from
splinter import Browser
# Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να πάρουμε τον κώδικα html της ιστοσελίδας
from bs4 import BeautifulSoup
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε, σε αρχείο csv import csv
```

```
# Ορίζουμε το όνομα χρήστη και τον κωδικό για να συνδεθούμε
username = ‘το όνομα χρήστη μας’ password = 'τον κωδικό
μας'
# Επιλέγουμε το browser (αφήνουμε τον προεπιλεγμένο - Firefox) browser
= Browser()
# Συμπληρώνουμε τη διεύθυνση της σελίδας browser.visit('https://www.goodreads.com/')
# Βρίσκουμε το πεδίο της φόρμας που χρησιμοποιείται για το όνομα χρήστη και το συμπληρώνουμε
με το όνομα που δώσαμε πιο πάνω
browser.find_by_id('userSignInFormEmail').first.fill(username)
# Αντίστοιχα βρίσκουμε το πεδίο του κωδικού και το συμπληρώνουμε
browser.find_by_id('user_password').first.fill(password)
# Βρίσκουμε το κουμπί που στέλνει τα δεδομένα για να συνδεθούμε και προσομοιώνουμε το
πάτημα. browser.find_by_value('Sign in').first.click()
# Όρίζουμε τον χρήστη από τον οποίο θα αρχίσει η καταγραφή των χρηστών url
= 'https://www.goodreads.com/friend/user/1713956-manny?page='
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τα δεδομένα usernames
= [[0 for i in range(1)]for j in range(1)]
# Boolean μεταβλητή η οποία μας επιβεβαιώνει ότι υπάρχει επόμενη σελίδα haveNext
= True
# Μετρητής για τον προσδιορισμό της σελίδας που θα επεξεργαστούμε i
= 1
# Κλήση της πρώτης σελίδας που θέλουμε να επεξεργαστούμε στο φυλλομετρητή  browser.visit(url
+ str(i))
# Επαναληπτική διαδικασία όσο υπάρχει επόμενη σελίδα while
haveNext:
# Κλήση της σελίδας που θέλουμε να επεξεργαστούμε στο φυλλομετρητή
browser.visit(url + str(i))
# Λήψη html κώδικα της προς επεξεργασία σελίδας
myHTML = browser.html
# Μετατρέπουμε τον html κώδικα στη μορφή που χρειάζεται το BeautifulSoup για να το
επεξεργαστεί     soup = BeautifulSoup(myHTML, 'html.parser')
# Ελέγχουμε αν υπάρχει επόμενη σελίδα
if not soup.find_all(rel='next'):
# Αν δεν υπάρχει επόμενη σελίδα, ενημερώνουμε τη Boolean μεταβλητή         haveNext = False
# Βρίσκουμε όλα τα σημεία το html κώδικα με το rel='acquaintance' ώστε να πάρουμε το url των
χρηστών της σελίδας
   unames = soup.find_all(rel='acquaintance')
# Ξεκινάμε επαναληπτική διαδικασία για κάθε ένα url που έχουμε στη λίστα
for nam in unames:
# Χωρίζουμε το url ώστε να πάρουμε το id του χρήστη
      userId = str.split(nam['href'], '/')
# Χωρίζουμε το id ώστε να πάρουμε μόνο το όνομα του χρήστη
      uname = str.split(userId[3], '-')
```

```
# Διαγράφω το πρώτο στοιχείο του πιο πάνω πίνακα καθώς δεν το χρειαζόμαστε
del uname[0]
# διαμορφώνουμε το τελικό όνομα του χρήστη με κενό ανάμεσα στις λέξεις
fUname = " ".join(uname)
# Προσθέτουμε το όνομα του χρήστη στην τελική του μορφή, στη λίστα με τα ονόματα
usernames.append([fUname, userId[3]])
# Εκτυπώνουμε στην οθόνη τον αριθμό της σελίδας που επεξεργαστήκαμε
print(str(i) + ' page dumped.')
# Αυξάνουμε το μετρητή σελίδας κατά ένα
i += 1
# Ανοίγουμε το αρχείο csv που θα αποθηκεύσουμε την πληροφορία
with open("users.csv", "w", newline='') as f:     writer = csv.writer(f)
# Γράφουμε τη λίστα με τα ονόματα των χρηστών στο αρχείο
writer.writerows(usernames)
```

- Κώδικας ανάκτησης φίλων χρηστών

```
# Εισαγωγή βιβλιοθήκης φυλλομετρητή
from splinter import Browser

# Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να πάρουμε τον κώδικα html της ιστοσελίδας
from bs4 import BeautifulSoup

# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε, σε αρχείο csv import csv

# Ορίζουμε το όνομα χρήστη και τον κωδικό για να συνδεθούμε username
= ‘το όνομα χρήστη μας’
password = 'τον κωδικό μας'

# Επιλέγουμε το browser (αφήνουμε τον προεπιλεγμένο - Firefox) browser
= Browser()

# Συμπληρώνουμε τη διεύθυνση της σελίδας browser.visit('https://www.goodreads.com/')

# Βρίσκουμε το πεδίο της φόρμας που χρησιμοποιείται για το όνομα χρήστη και το συμπληρώνουμε με το όνομα που δώσαμε πιο πάνω
browser.find_by_id('userSignInFormEmail').first.fill(username)

# Αντίστοιχα βρίσκουμε το πεδίο του κωδικού και το συμπληρώνουμε
browser.find_by_id('user_password').first.fill(password)

# Βρίσκουμε το κουμπί που στέλνει τα δεδομένα για να συνδεθούμε και προσμοιώνουμε το πάτημα.
browser.find_by_value('Sign in').first.click()
```

```
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τα δεδομένα links
= [[0 for i in range(1)]for j in range(1)]

# Ανοίγουμε το αρχείο csv που θα διαβάσουμε την πληροφορία csvReader
= csv.reader(open('users.csv', 'r'), delimiter=',');
# Διαβάζουμε τα ονόματα των χρηστών και τα κρατάμε στη λίστα που δημιουργήσαμε
for row in csvReader:     links.append(row);

# Διαγράφουμε το πρώτο στοιχείο καθώς είναι κενό links.pop(0)
# Ορίζουμε τη βασική διεύθυνση
mainUrl = 'https://www.goodreads.com/friend/user/'

for link in links:
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τα δεδομένα
usernames = [[0 for i in range(1)] for j in range(1)]
# Μετρητής για τον προσδιορισμό της σελίδας που θα επεξεργαστούμε
i = 1
# Boolean μεταβλητή η οποία μας επιβεβαιώνει ότι υπάρχει επόμενη σελίδα
haveNext = True
# Σχηματίζουμε τη διεύθυνση που θα επεξεργαστούμε
   url = mainUrl + link[1] + "?page="
# Επαναληπτική διαδικασία όσο υπάρχει επόμενη σελίδα
while haveNext:
# Κλήση της σελίδας που θέλουμε να επεξεργαστούμε στο φυλλομετρητή
browser.visit(url + str(i))
# Λήψη html κώδικα της προς επεξεργασία σελίδας
myHTML = browser.html
# Μετατρέπουμε τον html κώδικα στη μορφή που χρειάζεται το BeautifulSoup για να το
επεξεργαστεί
      soup = BeautifulSoup(myHTML, 'html.parser')
# Ελέγχουμε αν υπάρχει επόμενη σελίδα
if not soup.find_all(rel='next'):
# Αν δεν υπάρχει επόμενη σελίδα, ενημερώνουμε τη Boolean μεταβλητή
         haveNext = False
# Βρίσκουμε όλα τα σημεία το html κώδικα με το rel='acquaintance' ώστε να πάρουμε το url των
χρηστών της σελίδας
      unames = soup.find_all(rel='acquaintance')
# Ξεκινάμε επαναληπτική διαδικασία για κάθε ένα url που έχουμε στη λίστα
for nam in unames:
# Χωρίζουμε το url ώστε να πάρουμε το id του χρήστη
         userId = str.split(nam['href'], '/')
# Χωρίζουμε το id ώστε να πάρουμε μόνο το όνομα του χρήστη
         uname = str.split(userId[3], '-')
```

```
# Διαγράφω το πρώτο στοιχείο του πιο πάνω πίνακα καθώς δεν το χρειαζόμαστε
del uname[0]
# διαμορφώνουμε το τελικό όνομα του χρήστη με κενό ανάμεσα στις λέξεις
fUname = " ".join(uname)
# Προσθέτουμε το όνομα του χρήστη στην τελική του μορφή, στη λίστα με τα ονόματα
usernames.append([fUname, userId[3]])

# Εκτυπώνουμε στην οθόνη τον αριθμό της σελίδας που επεξεργαστήκαμε
print(str(i) + ' page dumped.')
# Αυξάνουμε το μετρητή σελίδας κατά ένα
i += 1

# Ορίζουμε το όνομα του αρχείου στο οποίο θα αποθηκεύσουμε τις πληροφορίες που διαβάσαμε
saveFile = "friends/" + link[0] + ".csv"

# Ανοίγουμε το αρχείο csv που θα αποθηκεύσουμε την πληροφορία
with open(saveFile, "w", newline='') as f:        writer = csv.writer(f)
# Γράφουμε τη λίστα με τα ονόματα των χρηστών στο αρχείο
writer.writerows(usernames)
```

- Κώδικας ανάκτησης βιβλίων χρηστών

```
# Εισαγωγή βιβλιοθήκης φυλλομετρητή from
splinter import Browser

# Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να πάρουμε τον κώδικα html της ιστοσελίδας
from bs4 import BeautifulSoup

# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε, σε αρχείο csv import csv

# Ορίζουμε το όνομα χρήστη και τον κωδικό για να συνδεθούμε username
= ‘το όνομα χρήστη μας’
password = 'τον κωδικό μας'


# Επιλέγουμε το browser (αφήνουμε τον προεπιλεγμένο - Firefox) browser
= Browser()

# Συμπληρώνουμε τη διεύθυνση της σελίδας browser.visit('https://www.goodreads.com/')
```

```
# Βρίσκουμε το πεδίο της φόρμας που χρησιμοποιείται για το όνομα χρήστη και το συμπληρώνουμε με το όνομα που δώσαμε πιο πάνω
browser.find_by_id('userSignInFormEmail').first.fill(username)

# Αντίστοιχα βρίσκουμε το πεδίο του κωδικού και το συμπληρώνουμε
browser.find_by_id('user_password').first.fill(password)

# Βρίσκουμε το κουμπί που στέλνει τα δεδομένα για να συνδεθούμε και προσμοιώνουμε το πάτημα.
browser.find_by_value('Sign in').first.click()

# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τα δεδομένα
links = [[0 for i in range(1)]for j in range(1)]

# Ανοίγουμε το αρχείο csv που θα διαβάσουμε την πληροφορία csvReader
= csv.reader(open('users2.csv', 'r'), delimiter=',');
# Διαβάζουμε τα ονόματα των χρηστών και τα κρατάμε στη λίστα που δημιουργήσαμε
for row in csvReader:
links.append(row);

# Διαγράφουμε το πρώτο στοιχείο καθώς είναι κενό links.pop(0)
# Ορίζουμε τη βασική διεύθυνση
mainUrl = 'https://www.goodreads.com/review/list/'

for link in links:
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τον τίτλο του βιβλίου
   tmpTitle = [[0 for i in range(1)] for j in range(1)]
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τη βαθμολογία του βιβλίου
tmpRating = [[0 for i in range(1)] for j in range(1)]
# Δημιουργούμε τη λίστα στην οποία θα αποθηκεύσουμε τα βιβλία του χρήστη
userBooks = [[0 for i in range(1)] for j in range(1)]
# Μετρητής για τον προσδιορισμό της σελίδας που θα επεξεργαστούμε
i = 1
# Boolean μεταβλητή η οποία μας επιβεβαιώνει ότι υπάρχει επόμενη σελίδα
haveNext = True
# Σχηματίζουμε τη διεύθυνση που θα επεξεργαστούμε
url = mainUrl + link[1] + "?page="
# Επαναληπτική διαδικασία όσο υπάρχει επόμενη σελίδα
while haveNext:
# Κλήση της σελίδας που θέλουμε να επεξεργαστούμε στο φυλλομετρητή
browser.visit(url + str(i) + "?shelf=read&sort=date_added")
```

```
# Λήψη html κώδικα της προς επεξεργασία σελίδας
myHTML = browser.html
# Μετατρέπουμε τον html κώδικα στη μορφή που χρειάζεται το BeautifulSoup για να το
επεξεργαστεί        soup = BeautifulSoup(myHTML, 'html.parser')
# Βρίσκουμε τον τίτλο του βιβλίου
      ubooks = soup.find_all("td", {"class": "field title"})
# Ξεκινάμε επαναληπτική διαδικασία για κάθε ένα βιβλίο που έχουμε στη λίστα
for boo in ubooks:
# Προσθέτουμε τον τίτλο του βιβλίου στη λίστα
          tmpTitle.append(boo.find('a')['title'])
# Βρίσκουμε τη βαθμολογία του βιβλίου
      urating = soup.find_all("span", {"class": " staticStars"})
# Ξεκινάμε επαναληπτική διαδικασία για κάθε βαθμολογία που έχουμε στη λίστα
for rate in urating:
          rat = str(rate).count("staticStar p10")
# Προσθέτουμε τη βαθμολογία του βιβλίου στη λίστα
tmpRating.append(rat)
# Ελέγχουμε αν υπάρχει επόμενη σελίδα
if not soup.find_all(rel='next'):
# Αν δεν υπάρχει επόμενη σελίδα, ενημερώνουμε τη Boolean μεταβλητή
haveNext = False
# Προσθέτουμε τον τίτλο του βιβλίου και τη βαθμολογία στη λίστα του κάθε χρήστη
for q in range(tmpTitle.__len__()):
              userBooks.append([tmpTitle[q],[tmpRating[q]]])

# Εκτυπώνουμε στην οθόνη τον αριθμό της σελίδας που επεξεργαστήκαμε
print(str(i) + ' page dumped.')
# Αυξάνουμε το μετρητή σελίδας κατά ένα
i += 1

# Ορίζουμε το όνομα του αρχείου στο οποίο θα αποθηκεύσουμε τις πληροφορίες που διαβάσαμε
saveFile = "booksReviews/" + link[0] + ".csv"

# Ανοίγουμε το αρχείο csv που θα αποθηκεύσουμε την πληροφορία
with open(saveFile, "w", newline='', encoding='utf-8') as f:
writer = csv.writer(f)
# Γράφουμε τη λίστα με βιβλία και τις βαθμολογίες των χρηστών στο αρχείο
writer.writerows(userBooks)
```

- Εισαγωγή χρηστών στη βάση

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε,
σε αρχείο csv import csv

# Συνδεόμαστε με τη βάση δεδομένων connection =
pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
                    password = 'τον κωδικό μας',
                    db='goodreads',
                    cursorclass=pymysql.cursors.DictCursor)

# Ανοίγουμε το αρχείο για να το διαβάσουμε with
open("users.csv") as users:
# Αποθηκεύουμε τα δεδομένα του αρχείου σε λίστα
myusers = csv.reader(users, delimiter=',')
# Επαναληπτική διαδικασία για κάθε χρήστη στη λίστα myusers
for user in myusers:        try:
# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση
with connection.cursor() as cursor:
# Ορίζουμε το sql ερώτημα που θέλουμε να εκτελέσουμε
        sql = "INSERT INTO `users` (`userURL`, `userName`) VALUES (%s, %s)"
# Καταχωρίζουμε στη βάση τα δύο στοιχεία που θέλουμε να συνδέσουμε
        cursor.execute(sql, (user[1], user[0]))

# Αποθηκεύουμε τις αλλαγές στη βάση
connection.commit()

# Αντιμετωπίζουμε το πιθανό σφάλμα που μπορεί να προκύψει
except:
        continue

# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

- Σύνδεση χρηστών με το κριτήριο της φιλίας

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε,
σε αρχείο csv import csv # Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να ανοίξουμε το
αρχείο από το λειτουργικό σύστημα import glob
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να φτιάξουμε το path από τις μεταβλητές import
os
```

```
# Ορίζουμε το φάκελο στον υπολογιστή μας από τον οποίο θα ανοίξουμε τα αρχεία path
= 'friends/'

# Συνδεόμαστε με τη βάση δεδομένων connection =
pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
password = 'τον κωδικό μας',
                        db='goodreads',
                        cursorclass=pymysql.cursors.DictCursor)

# Επαναληπτική διαδικασία για κάθε αρχείο στο φάκελο
for file in glob.glob(os.path.join(path, '*.csv')):     name =
file.split('/')
# Από το όνομα το αρχείου αντλούμε το όνομα του χρήστη.
   name = name[1].split('.')

try:
# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση
with connection.cursor() as cursor:
# Αναζητάμε στη βάση δεδομένων τη διεύθυνση του χρήστη            sql
= "SELECT userURL FROM users WHERE userName=%s"
cursor.execute(sql, name[0])
# Αποθηκεύουμε το url που μας επέστρεψε το ερώτημα
userURL = cursor.fetchone().get('userURL')

# Ανοίγουμε το αρχείο για να το διαβάσουμε
                            with open(file) as friends:
# Αποθηκεύουμε τα δεδομένα του αρχείου σε λίστα
            myfriends = csv.reader(friends, delimiter=',')
# Επαναληπτική διαδικασία για κάθε φίλο στη λίστα myfriends
for friend in myfriends:                    try:
# Αναζητάμε στη βάση δεδομένων τη σύνδεση του χρήστη με το φίλο
                    sql = "SELECT `userID`, `friendID` FROM `user_friends` WHERE (`userID`=%s)
AND (`friendID`=%s)"
                    cursor.execute(sql, (friend[1], userURL))
# Αν το ερώτημα δεν μας επιστρέψει αποτέλεσμα, καταχωρίζουμε τη σύνδεση στον πίνακα
χρηστών - φίλων                    if not cursor.fetchall():
                        sql = "INSERT INTO `user_friends` (`userID`, `friendID`) VALUES (%s, %s)"
cursor.execute(sql, (userURL, friend[1]))

                        connection.commit()
# Εκτυπώνουμε ενημερωτικό μήνυμα στην οθόνη
                        print("Inserted " + userURL + " " + friend[1])
else:
```

```
                    continue
except:
                continue
except:     continue
# Κλείνουμε το cursor
  cursor.close()

# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

- Εισαγωγή βιβλίων στη βάση

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε,
σε αρχείο csv import csv # Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να ανοίξουμε το
αρχείο από το λειτουργικό σύστημα import glob
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να φτιάξουμε το path από τις μεταβλητές import
os
# Ορίζουμε το φάκελο στον υπολογιστή μας από τον οποίο θα ανοίξουμε τα αρχεία path
= 'booksReviews/'
# Συνδεόμαστε με τη βάση δεδομένων connection =
pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
                    password = 'τον κωδικό μας',
                    db='goodreads',
                    cursorclass=pymysql.cursors.DictCursor)
# Επαναληπτική διαδικασία για κάθε αρχείο στο φάκελο
for file in glob.glob(os.path.join(path, '*.csv')):     name =
file.split('/')
# Από το όνομα το αρχείου αντλούμε το όνομα του χρήστη.
name = name[1].split('.')
# Ανοίγουμε το αρχείο για να το διαβάσουμε
with open(file) as books:
# Αποθηκεύουμε τα δεδομένα του αρχείου σε λίστα
mybooks = csv.reader(books, delimiter=',')
# Επαναληπτική διαδικασία για κάθε βιβλίο στη λίστα mybooks
for book in mybooks:            try:
# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση
with connection.cursor() as cursor:
# Αναζητάμε στη βάση δεδομένων το bookName (όνομα του βιβλίου) για να ελέγξουμε αν υπάρχει
το βιβλίο ήδη στη βάση
            sql = "SELECT bookName FROM books WHERE bookName=%s"
cursor.execute(sql, book[0]) # Αν το ερώτημα δεν μας επιστρέψει τίποτα
            if not cursor.fetchone():
```

```
# Εκτυπώνουμε ενημερωτικό μήνυμα στην οθόνη
                print("Inserting " + book[0])
try:
# Εισάγουμε το βιβλίο στη βάση
                    sql = "INSERT INTO `books` (`bookName`) VALUES (%s)"
cursor.execute(sql, (book[0]))                    connection.commit()
# Αντιμετωπίζουμε το πιθανό σφάλμα που μπορεί να προκύψει
except:                    continue
# Αντιμετωπίζουμε το πιθανό σφάλμα που μπορεί να προκύψει
except:
            continue
# Κλείνουμε το cursor
cursor.close()
# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

- Σύνδεση χρηστών με βιβλία που έχουν διαβάσει

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να αποθηκεύσουμε τα δεδομένα που θα πάρουμε,
σε αρχείο csv import csv # Εισαγωγή βιβλιοθήκης που θα χρησιμοποιήσουμε για να ανοίξουμε το
αρχείο από το λειτουργικό σύστημα import glob
# Εισαγωγή βιβλιοθήκης που θα μας βοηθήσει να φτιάξουμε το path από τις μεταβλητές import
os

# Ορίζουμε το φάκελο στον υπολογιστή μας από τον οποίο θα ανοίξουμε τα αρχεία path
= 'booksReviews/'

# Συνδεόμαστε με τη βάση δεδομένων connection =
pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
                      password = 'τον κωδικό μας',
                    db='goodreads',
                    cursorclass=pymysql.cursors.DictCursor)

# Επαναληπτική διαδικασία για κάθε αρχείο στο φάκελο for
file in glob.glob(os.path.join(path, '*.csv')):
   name = file.split('/')
# Από το όνομα το αρχείου αντλούμε το όνομα του χρήστη.
   name = name[1].split('.')

# Ανοίγουμε το αρχείο για να το διαβάσουμε
with open(file) as books:
```

```
# Αποθηκεύουμε τα δεδομένα του αρχείου σε λίστα
mybooks = csv.reader(books, delimiter=',')
# Επαναληπτική διαδικασία για κάθε βιβλίο στη λίστα mybooks
for book in mybooks:

try:
# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση
with connection.cursor() as cursor:
# Αναζητάμε στη βάση δεδομένων το userURL του χρήστη που είναι μοναδικό για τον κάθε χρήστη
sql = "SELECT userURL FROM users WHERE userName=%s"                cursor.execute(sql,
name[0])
                userURL = cursor.fetchone().get('userURL')

# Αναζητάμε στη βάση δεδομένων το bookID το οποίο είναι το στοιχείο που χαρακτηρίζει μοναδικά
το κάθε βιβλίο
                sql = "SELECT bookID FROM books WHERE bookName=%s"
cursor.execute(sql, book[0])
                bookID = cursor.fetchone().get('bookID')

# Αν μας έχουν επιστρέψει δεδομένα και για το userURL και για το bookID
if userURL and bookID:
# Εκτυπώνουμε ενημερωτικό μήνυμα στην οθόνη
                print("Inserting " + book[0] + " to user " + name[0] + " Rating " + book[1][1])

# Ορίζουμε το sql ερώτημα που θέλουμε να εκτελέσουμε
                sql = "INSERT INTO `user_books` (`userID`, `bookID`, `bookRating`) VALUES (%s,
%s, %s)"
# Καταχωρίζουμε στη βάση τα δύο στοιχεία που θέλουμε να συνδέσουμε
cursor.execute(sql, (userURL, bookID, book[1][1]))

# Αποθηκεύουμε τις αλλαγές στη βάση
                connection.commit()


# Αντιμετωπίζουμε το πιθανό σφάλμα που μπορεί να προκύψει
except:
            continue

# Κλείνουμε το cursor
cursor.close()

# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

- Εισαγωγή μέσου όρου

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors

# Συνδεόμαστε με τη βάση δεδομένων connection =
pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
                    password = 'τον κωδικό μας',
                    db='goodreads',
                    cursorclass=pymysql.cursors.DictCursor)


# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση with
connection.cursor() as cursor:
# Αναζητάμε στη βάση δεδομένων όλα τα urls των χρηστών
sql = "SELECT userURL FROM goodreads.users"
cursor.execute(sql)

# Αποθηκεύουμε το url που μας επέστρεψε το ερώτημα
   usersList = list(cursor.fetchall())

# Επαναληπτική διαδικασία για κάθε χρήστη στη λίστα usersList
for user in usersList:        try:
# Αναζητάμε στη βάση δεδομένων των αριθμό των βιβλίων και το σύνολο της βαθμολογίας που έχει
δώσει ο χρήστης
         sql = "SELECT count(userID) as BooksRead, sum(bookRating) as RatingSum FROM
goodreads.user_books WHERE userID = %s AND bookRating > 0; "
cursor.execute(sql, user["userURL"])

# Αποθηκεύουμε τα δεδομένα που μας επέστρεψε το ερώτημα σε λίστα
ratingSet = cursor.fetchone()
# Αρχικοποιούμε τη μεταβλητή στην οποία θα αποθηκεύσουμε το μέσο όρο της βαθμολογίας
userAverageRating = 0

# βρίσκουμε το μέσο όρο
         if ratingSet["BooksRead"] > 0 and ratingSet["RatingSum"] > 0:
            userAverageRating = ratingSet["RatingSum"] / ratingSet["BooksRead"]

try:
# Εκτελούμε εντολή update για να αποθηκεύσουμε το μέσο όρο στη βάση
            updateAverage = "UPDATE goodreads.users SET BooksRated = %s, AverageRating =
%s WHERE userURL = %s;"
            cursor.execute(updateAverage, (ratingSet["BooksRead"], userAverageRating,
user["userURL"]))

            connection.commit()
```

```
# Εκτυπώνουμε ενημερωτικό μήνυμα στην οθόνη
            print("Ok for user: ", user["userURL"])

except:
            continue

      except:
         continue

# Κλείνουμε το cursor
   cursor.close()

# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

- Εισαγωγή βάρους

```
# Εισαγωγή βιβλιοθήκης driver για τη σύνδεση με τη βάση δεδομένων mysql import
pymysql.cursors
import math

# Συνδεόμαστε με τη βάση δεδομένων connection
= pymysql.connect(host='localhost',
username = ‘το όνομα χρήστη μας’,
                     password = 'τον κωδικό μας'
                  db='goodreads',
                  cursorclass=pymysql.cursors.DictCursor)

# Δημιουργούμε ένα cursor που θα μας βοηθήσει να εκτελέσουμε ερωτήματα στη βάση with
connection.cursor() as cursor:
# Αναζητάμε στη βάση δεδομένων τους χρήστες με τους φίλους τους
   sql = "SELECT goodreads.user_friends.userID, goodreads.user_friends.friendID FROM
goodreads.user_friends order by goodreads.user_friends.userID;"     cursor.execute(sql)

# Αποθηκεύουμε τα αποτελέσματα του ερωτήματος σε λίστα
   usersList = list(cursor.fetchall())
# Επαναληπτική διαδικασία για κάθε βιβλίο στη λίστα usersList
for user in usersList:        try:
         sql = "select goodreads.user_books.bookID from goodreads.user_books where
goodreads.user_books.userID = %s;"
# Αναζητάμε στη βάση δεδομένων τα id των βιβλίων που έχει ο κάθε χρήστης στη λίστα του
cursor.execute(sql, user["userID"])

# Αποθηκεύουμε σε πίνακα όλα τα αποτελέσματα που πήραμε για τα βιβλία του χρήστη
userBooks = [item['bookID'] for item in cursor.fetchall()]
```

```
# Αναζητάμε στη βάση δεδομένων τα id των βιβλίων που έχει ο φίλος του πιο πάνω χρήστη στη λίστα του
        cursor.execute(sql, user["friendID"])

# Αποθηκεύουμε σε πίνακα όλα τα αποτελέσματα που πήραμε για τα βιβλία του φίλου του χρήστη
        friendBooks = [item['bookID'] for item in cursor.fetchall()]

# Μετατρέπουμε τους πίνακες σε sets
        sa = set(userBooks)
        sb = set(friendBooks)

# Βρίσκουμε τα κοινά βιβλία μεταξύ του χρήστη και του φίλου του
        c = sa.intersection(sb)

# Αρχικοποιούμε τις μεταβλητές που θα χρειαστούμε για να αποθηκεύσουμε το σύνολο της βαθμολογίας για κάθε χρήστη και δύο βοηθητικές που θα τις χρησιμοποιήσουμε για να υπολογίσουμε το βάρος μεταξύ των δύο χρηστών ratingSum = 0
squareRootUser = 0
        squareRootFriend = 0

# Επαναληπτική διαδικασία για κάθε κοινό βιβλίο στη λίστα c
for book in c:
            try:
                sql = "SELECT goodreads.user_books.bookRating FROM goodreads.user_books where goodreads.user_books.userID = %s and goodreads.user_books.bookID = %s;" # Αναζητάμε στη βάση δεδομένων τη βαθμολογία που έχει δώσει ο χρήστης στο βιβλίο
                        cursor.execute(sql, (user["userID"], book))

# Αποθηκεύουμε σε λίστα τη βαθμολογία του χρήστη
userBooksRating = cursor.fetchone()

# Αναζητάμε στη βάση δεδομένων τη βαθμολογία που έχει δώσει ο φίλος του χρήστη στο βιβλίο
cursor.execute(sql, (user["friendID"], book))
# Αποθηκεύουμε σε λίστα τη βαθμολογία του φίλου του χρήστη
friendBooksRating = cursor.fetchone()


# Ελέγχουμε αν η βαθμολογία είναι 0 ωστε να μη τη λάβουμε υπόψη μας if (userBooksRating['bookRating'] == 0) or (friendBooksRating['bookRating'] == 0):
continue else:
                ratingSum += userBooksRating['bookRating'] * friendBooksRating['bookRating']
squareRootUser += math.sqrt(userBooksRating['bookRating'] * userBooksRating['bookRating'])
                squareRootFriend += math.sqrt(friendBooksRating['bookRating'] * friendBooksRating['bookRating'])
```

```
except:
                continue
# Βρίσκουμε το βάρος μεταξύ των δύο φίλων            userFriendWeight =
ratingSum / (squareRootUser * squareRootFriend)            try:
# Κάνουμε update την εγγραφή με τους δύο φίλους ώστε να αποθηκεύσουμε και το βάρος τους
sql = "UPDATE goodreads.user_friends SET goodreads.user_friends.Weight = %s where
goodreads.user_friends.userID = %s and goodreads.user_friends.friendID = %s"
            cursor.execute(sql, (round(userFriendWeight, 10), user["userID"], user["friendID"]))

            connection.commit()

except:
            continue
# Εκτυπώνουμε ενημερωτικό μήνυμα στην οθόνη
          print(user["userID"], " - ", user["friendID"], " = ", userFriendWeight)

       except:
          continue

# Κλείνουμε το cursor
   cursor.close()

# Κλείνουμε τη σύνδεση προς τη βάση connection.close()
```

7. Βιβλιογραφικές αναφορές

Ξενόγλωσση Βιβλιογραφία

- Opinion Mining and Sentiment Analysis - Bo Pang and Lillian Lee Foundations and TrendsR in, Information Retrieval Vol. 2, Nos. 1–2 (2008) 1–135, 2008 B. Pang and L. Lee, DOI: 10.1561/1500000001
- Recommender Systems Handbook - Francesco Ricci · Lior Rokach · Bracha Shapira · Paul B. Kantor
  ISBN 978-0-387-85819-7 e-ISBN 978-0-387-85820-3
  DOI 10.1007/978-0-387-85820-3
  Springer New York Dordrecht Heidelberg London
  Library of Congress Control Number: 2010937590
  c Springer Science+Business Media, LLC 2011
- Learn how to use Gephi – Gephi https://gephi.org/users/
- Scrapy 1.5 documentation - Scrapy https://docs.scrapy.org/en/latest/

Ελληνόγλωσση Βιβλιογραφία

- Συστήματα βάσεων δεδομένων - Abraham Silberscharz · Henry F. Korth · S. Sudarshan Εκδόσεις Μ. Γκιούρδας, 6η έκδοση, ISBN 978-960-512-623-0
- Περιγραφική και Διερευνητική Στατιστική. Ανάλυση Δεδομένων, τόμος Α΄ - Τσίμπος Κλέων · Γεωργιακώδης Φώτης
  Εκδόσεις Σταμούλη, χρονολογία έκδοσης 1999, ISBN 9603512427