2. Αλγόριθμοι Συνεργατικού Φιλτραρίσματος

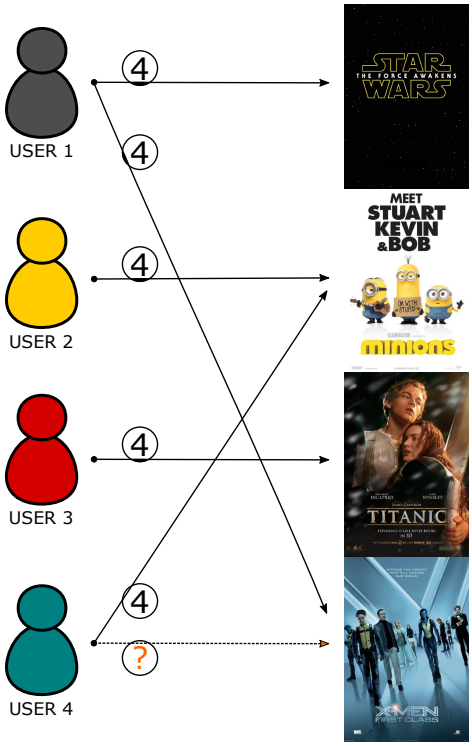
***Σύνοψη**: Στο δεύτερο κεφάλαιο θα εστιάσουμε στους **αλγορίθμους συνεργατικού φιλτραρίσματος (ΣΦ) βάσει των** k**-κοντινότερων γειτόνων**. Αρχικά, θα κάνουμε μια εκτενή περιγραφή και ανάλυση των παραγόντων που εμπλέκονται και επηρεάζουν το **ΣΦ (μέτρα ομοιότητας**, **τρόποι δημιουργίας γειτονιάς του υπό εξέταση χρήστη/στοιχείου** κ.λπ.). Στη συνέχεια, θα περιγράψουμε διεξοδικά τους τρόπους δημιουργίας της **λίστας προτεινόμενων στοιχείων**. Τέλος, θα παρουσιάσουμε συνοπτικά τους τρόπους μέτρησης της **αποτελεσματικότητας (μετρικές ακριβείας**, **ανάκτησης** κ.λπ.) των αλγορίθμων συστάσεων.*

***Προαπαιτούμενη γνώση**: Ο αναγνώστης καλό θα ήταν να έχει μια πρότερη επαφή με το αντικείμενο των αλγορίθμων **Ανάκτησης Πληροφορίας**.*

2.1 Εισαγωγή

Ο Παγκόσμιος Ιστός περιέχει δισεκατομμύρια σελίδες και αναπτύσσεται ραγδαία μέρα με τη μέρα. Υπάρχει μια τεράστια ποσότητα πληροφοριών, η οποία δυσκολεύει τους ανθρώπους στο να βρίσκουν αυτό που μπορεί να χρειάζονται ή επιθυμούν. Το **"Συνεργατικό Φιλτράρισμα" (ΣΦ) (Collaborative Filtering - CF)** είναι η πλέον επιτυχημένη μέθοδος στα συστήματα συστάσεων και τις εφαρμογές ηλεκτρονικού εμπορίου, διότι χρησιμοποιεί συνεργατικά τη γνώση και την εμπειρία του κοινωνικού συνόλου για το φιλτράρισμα στοιχείων και υπηρεσιών που ενδιαφέρουν τον υπό εξέταση χρήστη. Μ' αυτόν τον τρόπο άλλωστε αντιμετωπίζεται τα τελευταία χρόνια και το πρόβλημα της *υπερπληροφόρησης*. Στην καθημερινή μας ζωή, για την επιλογή των κατάλληλων για εμάς προϊόντων, ταινιών κ.λπ., συχνά δεχόμαστε συστάσεις από φίλους και γνωστούς. Στον Παγκόσμιο Ιστό, ωστόσο, η πληθώρα των διαθέσιμων προϊόντων καθιστά δύσκολη την εύρεση αυτών που θα μας ενδιέφεραν περισσότερο. Η λύση είναι να αποφύγουμε τις συστάσεις που γίνονται σε ατομικό επίπεδο και να περάσουμε σε αυτές που βασίζονται σε επίπεδο κοινωνικού συνόλου.

Κατόπιν θα εξετάσουμε την περίπτωση ενός απλού συστήματος σύστασης με τέσσερεις χρήστες που έχουν αξιολογήσει τέσσερεις ταινίες. Στο Σχήμα 2.1, λοιπόν, παρουσιάζεται ένας διμερής γράφος χρηστών-ταινιών όπου κάθε ακμή στο γράφημα μεταξύ ενός χρήστη και μιας ταινίας έχει ένα βάρος που δείχνει τον βαθμό προτίμησης του πρώτου στη δεύτερη - τονίζουμε ότι στο Σχήμα 2.1 παρουσιάζουμε μόνο "θετικές" αξιολογήσεις (>= 4) της κλίμακας βαθμολογίας 0-5. Ας υποθέσουμε στο τρέχον παράδειγμα ότι θέλουμε να προβλέψουμε τη βαθμολογία του χρήστη 4 για την ταινία *X-MEN*. Για να δείξουμε αυτήν την υπόθεση, χρησιμοποιούμε μια διακεκομμένη γραμμή για την ακμή που συνδέει τον χρήστη 4 με την ταινία 4, ενώ η πιθανή βαθμολογία εμφανίζεται μ' ένα ερωτηματικό. Η πρόκληση ενός συστήματος σύστασης είναι να προβλέψει σωστά μια βαθμολογία για τα στοιχεία εκείνα για τα οποία ένας χρήστης δεν έχει εκφράσει ρητά την προτίμησή του (δεν έχει βάλει βαθμολογία). Σε περίπτωση που το σύστημά μας προβλέψει τη βαθμολογία αυτή ως θετική (δηλαδή υψηλότερη από το 4 σε κλίμακα 0-5), τότε θα μπορούσε να προτείνει την ταινία *X-MEN* στον υπό εξέταση χρήστη 4.



Σχήμα 2.1: Τέσσερεις χρήστες έχουν βαθμολογήσει τέσσερεις ταινίες. Βάσει αυτών των αλληλεπιδράσεων ποια είναι η πρόβλεψή μας για τη βαθμολογία του χρήστη 4 στην ταινία *X-MEN*;

Για την παροχή συστάσεων που βασίζονται σε χρήστες με παρόμοιες προτιμήσεις το *ΣΦ* εφαρμόζει τεχνικές *Ανάκτησης Πληροφορίας, Εξόρυξης Δεδομένων* και *Μηχανικής Μάθησης*. Έχουν προταθεί δύο τύποι αλγορίθμων *ΣΦ*:

(α) Ο *αλγόριθμος k-κοντινότερων γειτόνων (k-nearest neighbors)*, ο οποίος στηρίζεται στη *δυναμική (on the fly)* δημιουργία μιας γειτονιάς για τον υπό εξέταση χρήστη που αποτελείται από τους κοντινότερους σ' αυτόν χρήστες βάσει κοινών τους προτιμήσεων μ' αυτόν.

(β) Οι *αλγόριθμοι που βασίζονται στη δημιουργία ενός μοντέλου (model-based algorithms)*, οι οποίοι αναπτύσσουν αρχικά ένα μοντέλο σχετικό με τις προτιμήσεις των χρηστών και, έπειτα, βάσει αυτού του μοντέλου κάνουν συστάσεις στον υπό εξέταση χρήστη.

Στο κεφάλαιο αυτό εστιάζουμε σε *αλγόριθμους ΣΦ βάσει των k-κοντινότερων γειτόνων*. Αρχικά παρέχουμε μια ενδελεχή ανάλυση των παραγόντων που εμπλέκονται στο ΣΦ. Το υπόλοιπο του κεφαλαίου οργανώνεται ως εξής: Η Ενότητα 2.2 συνοψίζει τις βασικές εργασίες σ' αυτό το ερευνητικό πεδίο, ενώ η Ενότητα 2.3 περιέχει την ανάλυση των εξετασθέντων παραγόντων *ΣΦ*.

2.2 Σχετική Βιβλιογραφία

Το πληροφοριακό σύστημα *Tapestry* [Goldberg et al., 1992] εφάρμοσε πρώτο το *ΣΦ* το 1992. Το 1994, ένα άλλο πληροφοριακό σύστημα, το *GroupLens* [Resnick et al., 1994] εφάρμοσε έναν αλγόριθμο *ΣΦ* βασισμένο σε προτιμήσεις χρηστών που σήμερα είναι γνωστός ως *αλγόριθμος ΣΦ βάσει χρήστη (User-based CF* ή αλλιώς, *UBCF)*, επειδή βρίσκει ομοιότητες χρηστών για να διαμορφώσει τη *γειτονιά των κοντινότερων εξ αυτών στον υπό εξέταση χρήστη*. Από τότε έχουν προταθεί πολλές αναβαθμίσεις του παραπάνω αλγορίθμου. Για παράδειγμα η εργασία των Herlocker και λοιπών συν-συγγραφέων [Herlocker et al., 1999], όπου η ομοιότητα μεταξύ δύο χρηστών πρέπει να υπολογίζεται μόνο εφόσον εξασφαλίζεται ένας ελάχιστος αριθμός στοιχείων τα οποία θα έχουν βαθμολογήσει και οι δύο. Με άλλα λόγια, οι δύο υπό εξέταση χρήστες θα πρέπει να έχουν αξιολογήσει έναν ελάχιστο αριθμό κοινών στοιχείων (π.χ. προϊόντα, ταινίες κ.λπ.).

Το 2001 προτάθηκε ένας ακόμη αλγόριθμος *ΣΦ* ο οποίος βασίζεται στις ομοιότητες στοιχείων για τη δημιουργία γειτονιάς ([Karypis, 2001], [Sarwar et al., 2001]). Είναι γνωστός ως *αλγόριθμος ΣΦ βάσει στοιχείου (Item-based CF* ή αλλιώς, *IBCF*), διότι δημιουργεί μια *γειτονιά k-κοντινότερων στοιχείων* αντί μιας *k-κοντινότερων χρηστών*.

Οι πιο πρόσφατες εργασίες ακολούθησαν τις δύο προαναφερθείσες κατευθύνσεις (δηλαδή *ΣΦ βάσει χρήστη* και *ΣΦ βάσει στοιχείου*). Ενδεικτικά, σε μία από αυτές οι συγγραφείς [Herlocker et al., 2002] σταθμίζουν τις ομοιότητες βάσει ενός ελαχίστου αριθμού κοινών αξιολογήσεων μεταξύ χρηστών/στοιχείων. Ενώ σε άλλη [Xue et al., 2005], προτείνεται μια υβριδική συνένωση των προαναφερθέντων αλγορίθμων με τους αλγορίθμους *ΣΦ βάσει μοντέλου*. Στη συνέχεια θα αναφερθούμε περαιτέρω στη σχετική βιβλιογραφία [Symeonidis et al., 2006] μέσω της ανάλυσης των εξεταζομένων παραγόντων που επηρεάζουν το *ΣΦ*.

2.3 Παράγοντες και Προκλήσεις στο ΣΦ

Στο σημείο αυτό θα αναφέρουμε τους κύριους παράγοντες που επηρεάζουν όλους τους *αλγορίθμους ΣΦ* και των οποίων η επίλυση αποτελεί τεχνική πρόκληση. Η ανάλυσή μας θα εστιάσει στις βασικές λειτουργίες του *ΣΦ*, που αποτελείται από τρία στάδια:

1. **Διαμόρφωση της γειτονιάς του υπό εξέταση χρήστη ή στοιχείου**.

2. **Δημιουργία της λίστας συστάσεων (Recommendation List)**.

3. **Αποτίμηση της λίστας συστάσεων**.

Οι παράγοντες που επηρεάζουν το *ΣΦ* συνοψίζονται παρακάτω, στον Πίνακα 2.1:

Όνομα Παράγοντα	**Σύντομη Περιγραφή**	**Στάδιο**
Αραιότητα δεδομένων	Όταν το ποσοστό αξιολογηθέντων προϊόντων είναι πολύ μικρό.	1
Κλιμάκωση	Η ανάγκη σε υπολογιστική ισχύ με την αύξηση χρηστών/στοιχείων.	1
Μέγεθος δεδομένων	Το μέγεθος των δεδομένων εκπαίδευσης/ελέγχου.	1, 3
Μέγεθος γειτονιάς	Ο αριθμός των γειτονικών χρηστών.	1
Μέτρο Ομοιότητας	Μέτρα για τον υπολογισμό της εγγύτητας χρηστών/στοιχείων.	1
Μέγεθος λίστας συστάσεων	Ο αριθμός των κορυφαίων-N προτεινόμενων στοιχείων.	2
Δημιουργία λίστας συστάσεων	Αλγόριθμοι για τη δημιουργία προτεινόμενων στοιχείων.	2
Κατώφλι θετικής βαθμολογίας	Ορίζει εάν είναι θετική ή αρνητική η αξιολόγηση ενός προϊόντος.	2, 3
Μετρικές Αποτίμησης	Μετρικές για την αποτίμηση των προτεινόμενων στοιχείων.	3
Ορισμός μεθόδου αναφοράς	Ορίζει μια μέθοδο αναφοράς για λόγους σύγκρισης.	3
Παρελθοντικά/μελλοντικά δεδομένα	Ο διαχωρισμός σε δεδομένα ελέγχου και δεδομένα εκπαίδευσης.	3

Πίνακας 2.1: Παράγοντες που επηρεάζουν το *ΣΦ*.

Παράδειγμα 2.1 Για να διευκολύνουμε την περιγραφή του *ΣΦ* θα χρησιμοποιήσουμε ένα παράδειγμα. Δίνεται ο πίνακας βαθμολογιών χρηστών-στοιχείων του Σχήματος 2.2, όπου U_{1-10} οι χρήστες, ενώ I_{1-6} τα στοιχεία που αυτοί έχουν βαθμολογήσει. Η κλίμακα βαθμολογίας είναι 1 έως 5. Όπως φαίνεται, τα δεδομένα του παραδείγματός μας είναι χωρισμένα σε δύο είδη, *εκπαίδευσης* και *ελέγχου*. Τα κενά κελιά δεν έχουν τιμή (δηλαδή είναι χωρίς βαθμολογία).

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4	1	1	4	-	-
U_2	1	4	4	-	-	4
U_3	2	1	4	-	-	4
U_4	1	2	1	1	-	-
U_5	4	1	2	-	1	-
U_6	1	5	4	-	4	-

(α)

	I_1	I_2	I_3	I_4	I_5	I_6
U_7	2	1	4	-	1	-
U_8	1	2	1	-	2	5
U_9	4	1	2	1	1	-
U_{10}	1	4	1	-	4	-

(β)

Σχήμα 2.2: (α) *Δεδομένα Εκπαίδευσης* ($n \times m$), (β) *Δεδομένα Ελέγχου*.

Στις επόμενες ενότητες αυτού του κεφαλαίου θα εμβαθύνουμε στους προαναφερθέντες παράγοντες *ΣΦ*, οι οποίοι αναλύονται ανάλογα με το στάδιο όπου ο καθένας εμπλέκεται. Ο Πίνακας 2.2 συνοψίζει τα σύμβολα που θα χρησιμοποιούνται στο εξής:

Σύμβολο	**Ορισμός**
k	αριθμός κοντινότερων γειτόνων
N	μέγεθος λίστας συστάσεων
P_τ	κατώφλι για θετικές βαθμολογίες
I	σύνολο όλων των στοιχείων
U	σύνολο όλων των χρηστών
u, v	μερικοί χρήστες
i, j	μερικά στοιχεία
I_u	σύνολο στοιχείων βαθμολογημένων από τον χρήστη u
U_i	σύνολο χρηστών που βαθμολόγησαν το στοιχείο i
$r_{u,i}$	η βαθμολογία του χρήστη u στο στοιχείο i
$\overline{r}_u$	μέση τιμή βαθμολογίας του χρήστη u
$\overline{r}_i$	μέση τιμή βαθμολογίας για το στοιχείο i
$p_{u,i}$	η προβλεφθείσα βαθμολογία του χρήστη u στο στοιχείο i

Πίνακας 2.2: Σύμβολα και Ορισμοί.

2.3.1 Παράγοντες Πρώτου Σταδίου

Αραιότητα δεδομένων. Είναι συχνό το φαινόμενο στα ηλεκτρονικά καταστήματα ή στα μέσα κοινωνικής δικτύωσης οι χρήστες να βαθμολογούν ένα πολύ μικρό ποσοστό στοιχείων (προϊόντα, αναρτήσεις χρηστών κ.λπ.), με συνέπεια τα δεδομένα να είναι πολύ "αραιά". Συνεπώς, λόγω έλλειψης επαρκών δεδομένων μπορεί να οδηγούμαστε σε ανακριβείς συστάσεις προϊόντων και υπηρεσιών. Για τον λόγο αυτό, ένας αριθμός εργασιών επικεντρώνει σε σύνολα δεδομένων (π.χ. *Movielens*) που είναι "αραιά" ([Herlocker et al., 2002], [Karypis, 2001], [Sarwar et al., 2001]). Ωστόσο, από την άλλη, υφίστανται σύνολα δεδομένων (πχ. *Jester*) που έχουν μεγαλύτερη πυκνότητα βαθμολογιών [Goldberg et al., 2001]. Συνεπώς, θα πρέπει να εξετάσουμε και τις δύο περιπτώσεις προκειμένου να εξαγάγουμε ολοκληρωμένα συμπεράσματα για την ποιότητα των αλγορίθμων συστάσεων.

Κλιμάκωση. Η δυνατότητα *κλιμάκωσης ενός αλγορίθμου συστάσεων* είναι σημαντική, επειδή στις πρακτικές εφαρμογές ο αριθμός χρηστών και στοιχείων είναι πολύ μεγάλος. Ένας τρόπος αντιμετώπισης του προβλήματος αυτού προτείνεται σε σχετική εργασία, [Sarwar et al., 2000b], όπου χρησιμοποιούνται τεχνικές *μείωσης των διαστάσεων* (*dimensionality reduction*) του *δισδιάστατου πίνακα βαθμολογιών χρηστών-στοιχείων*. Σημειώστε ότι αυτές οι τεχνικές είναι δυνατόν να αντισταθμίσουν την *αποτελεσματικότητα* με τον *χρόνο εκτέλεσης* των *αλγορίθμων ΣΦ*.

Μέγεθος δεδομένων εκπαίδευσης/ελέγχου. Υπάρχει σαφής αλληλεπίδραση μεταξύ του *μεγέθους των δεδομένων* εκπαίδευσης και της *ακρίβειας* και *αποτελεσματικότητας* των *αλγορίθμων ΣΦ* [Sarwar et al., 2001]. Από την άλλη μεριά, *μικρά μεγέθη δεδομένων* εκπαίδευσης επιδρούν αρνητικά στην *ακρίβεια* και την *αποτελεσματικότητα* των αλγορίθμων. Συνεπώς, η ακριβής αξιολόγηση των *αλγορίθμων ΣΦ* θα πρέπει να βασίζεται σε επαρκή ως προς το μέγεθος δεδομένα εκπαίδευσης. Παρότι οι περισσότερες έρευνες σχετικά με το παραπάνω θέμα χρησιμοποιούν περίπου το 80% των δεδομένων, υπάρχουν και εργασίες που αξιοποιούν σημαντικά μικρότερα μεγέθη [McLauglin and Herlocker, 2004]. Βάσει της σχετικής βιβλιογραφίας μπορούμε να συμπεράνουμε ότι ένα μέγεθος της τάξης του 75% του συνόλου των δεδομένων εκπαίδευσης αποτελεί μια επαρκή επιλογή.

Μέγεθος γειτονιάς. Ο αριθμός των k *πλησιεστέρων γειτόνων*, ο οποίος χρησιμοποιείται για τη *διαμόρφωση της γειτονιάς του υπό εξέταση χρήστη*, επιδρά άμεσα στην *ακρίβεια* και την *αποτελεσματικότητα* των *αλγορίθμων ΣΦ*. Σχετικές εργασίες ([Herlocker et al., 1999], [Sarwar et al., 2001]) μελετούν το k στο εύρος τιμών μεταξύ 10 και 100 για το σύνολο δεδομένων *Movielens 100K*. Το βέλτιστο k εξαρτάται από τα χαρακτηριστικά των δεδομένων. Συνεπώς, οι *αλγόριθμοι ΣΦ* θα πρέπει να αξιολογούνται έναντι κυμαινόμενου k. Επιπλέον, ένα ζήτημα που δεν έχει αποσαφηνισθεί σε σχετικές εργασίες είναι το εάν θα πρέπει να συμπεριλαμβάνουμε στη *γειτονιά του υπό εξέταση χρήστη* έναν χρήστη που σχετίζεται με *αρνητική ομοιότητα* με τον προαναφερθέντα. Με στόχο τη βελτίωση της *αποτελεσματικότητας* των *αλγορίθμων ΣΦ* προτείνουμε την τήρηση των θετικών μόνο ομοιοτήτων για τη διαμόρφωση γειτονιάς ακόμη και αν παραμείνουν λιγότεροι γείτονες από τον συγκεκριμένο αριθμό k.

Μέτρο ομοιότητας. Τα πιο δημοφιλή *μέτρα ομοιότητας* για τη *διαμόρφωση της γειτονιάς του υπό εξέταση χρήστη ή στοιχείου* βασίζονται στη *Συσχέτιση κατά Pearson* (*Pearson Correlation*) και στην *Ομοιότητα Συνημιτόνου* (*Cosine Similarity*) ([Herlocker et al., 2002], [McLauglin and Herlocker, 2004], [Sarwar et al., 2001]). Ειδικότερα, οι *αλγόριθμοι ΣΦ βάσει χρήστη* χρησιμοποιούν κυρίως τη *Συσχέτιση κατά Pearson* (Εξίσωση 2.1), ενώ για τους *αλγορίθμους ΣΦ βάσει στοιχείου* προτιμάται το *Μέτρο Ομοιότητας Προσαρμοσμένου Συνημιτόνου* (*Adjusted Cosine Similarity*) όπως φαίνεται στην Εξίσωση 2.2 ([McLauglin and Herlocker, 2004], [Sarwar et al., 2001]). Το δεύτερο είναι μια παραλλαγή του μέτρου *απλής Ομοιότητας Συνημιτόνου* που εξομαλύνει την πιθανή ύπαρξη *μεροληψίας* λόγω υποκειμενικών βαθμολογιών των χρηστών.

$$\mathrm{sim}(u,v) = \frac{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall i \in S} (r_{v,i} - \overline{r}_v)^2}}, \quad S = I_u \cap I_v. \tag{2.1}$$

$$\mathrm{sim}(i,j) = \frac{\sum_{\forall u \in T} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{\forall u \in U_i} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall u \in U_j} (r_{u,j} - \overline{r}_u)^2}}, \quad T = U_i \cap U_j. \tag{2.2}$$

Ο Herlocker και οι λοιποί συν-συγγραφείς [Herlocker et al., 2002] πρότειναν μια παραλλαγή των προαναφερθέντων μέτρων που σταθμίζει τις ομοιότητες από τον αριθμό των κοινών βαθμολογιών όταν αυτές είναι λιγότερες από ορισμένη παράμετρο κατωφλιού γ. Στο εξής, αυτό το μέτρο θα αναφέρεται με τον όρο *Σταθμισμένη Ομοιότητα* (*Weighted Similarity*) και θα δηλώνεται ως WS. Εάν λοιπόν sim είναι το *μέτρο ομοιότητας* υπολογισμένο είτε με την Εξίσωση 2.1 είτε με την Εξίσωση 2.2, τότε η WS ισούται με $\frac{\max(c,\gamma)}{\gamma}\ sim$ όπου c ο ελάχιστος αριθμός των στοιχείων που απαιτείται να έχουν βαθμολογήσει από κοινού οι δύο υπό εξέταση χρήστες.

Η *Συσχέτιση κατά Pearson* (Εξίσωση 2.1) λαμβάνει υπόψη μόνο το σύνολο των στοιχείων S, τα οποία έχουν *από κοινού βαθμολογηθεί* και από τους δύο χρήστες. Με την ίδια λογική, *το μέτρο Ομοιότητας Προσαρμοσμένου Συνημιτόνου* (Εξίσωση 2.2) εξετάζει στον αριθμητή μόνο το σύνολο των χρηστών T, οι οποίοι από κοινού βαθμολόγησαν και τα δύο ζεύγη στοιχείων. Αντιθέτως, ο παρονομαστής της Εξίσωσης 2.2 λαμβάνει υπόψη όλα τα στοιχεία που βαθμολόγησε κάθε χρήστης. Αξίζει να σημειωθεί ότι μόνο λίγες εργασίες ([McLauglin and Herlocker, 2004], [Sarwar et al., 2001]) είναι σαφείς ως προς τον προαναφερθέντα ορισμό για το S και το T. Ακολουθεί η εφαρμογή της *Συσχέτισης κατά Pearson* (Εξίσωση 2.1) στα δεδομένα του Παραδείγματος 2.1.

Παράδειγμα 2.2 Να υπολογίσετε τον πίνακα ομοιοτήτων χρηστών και τον πίνακα ομοιοτήτων στοιχείων για τα δεδομένα του παραδείγματος 2.1., καθώς και τους 3-κοντινότερους γείτονες του κάθε υπό εξέταση χρήστη και στοιχείου. Η εφαρμογή της *Συσχέτισης κατά Pearson* (Εξίσωση 2.1) στο τρέχον παράδειγμα του κεφαλαίου εμφανίζεται στο Σχήμα 2.3α, ενώ οι *k-κοντινότεροι γείτονες (k-NN)* στο Σχήμα 2.3β. Αντίστοιχα, οι υπολογισμένες ομοιότητες μεταξύ των στοιχείων βάσει της Εξίσωσης 2.2 παρουσιάζονται στο Σχήμα 2.3γ, ενώ στο Σχήμα 2.3δ βρίσκουμε τους *κοντινότερους γείτονες του κάθε υπό εξέταση στοιχείου*. Δεδομένου ότι λαμβάνονται υπόψη μόνο θετικές ομοιότητες μεταξύ χρηστών για τη *διαμόρφωση της γειτονιάς του κάθε υπό εξέταση χρήστη*, στους πίνακες υπάρχουν και κενά πεδία.

	U_1	U_2	U_3	U_4	U_5	U_6
U_7	-0.19	0.19	1	-0.76	0.33	-0.14
U_8	-0.5	0.44	0.38	1	-0.82	0.67
U_9	0.41	-0.94	0.14	-0.47	1	-0.95
U_{10}	-0.5	0.5	-0.76	1	-0.82	0.67

(α)

	$1^{st}NN$	$2^{st}NN$	$3^{st}NN$
U_7	$U_3(1)$	$U_5(0.33)$	$U_2(0.19)$
U_8	$U_4(1)$	$U_6(0.67)$	$U_2(0.44)$
U_9	$U_5(1)$	$U_1(0.41)$	$U_3(0.14)$
U_{10}	$U_4(1)$	$U_6(0.67)$	$U_2(0.5)$

(β)

	I_1	I_2	I_3	I_4	I_5	I_6
I_1	-	-0.65	-0.66	0.36	-0.68	-0.42
I_2	-0.65	-	0.18	-0.51	0.50	-0.36
I_3	-0.66	0.18	-	-0.66	0.10	0.67
I_4	0.36	-0.51	-0.66	-	0	0
I_5	-0.68	0.50	0.10	0	-	0
I_6	-0.42	-0.36	0.67	0	0	-

(γ)

	$1^{st}NN$	$2^{st}NN$	$3^{st}NN$
I_1	$I_4(0.36)$	-	-
I_2	$I_5(0.50)$	$I_3(0.18)$	-
I_3	$I_6(0.67)$	$I_2(0.18)$	$I_5(0.10)$
I_4	$I_1(0.36)$	-	-
I_5	$I_2(0.50)$	I3(0.10)	-
I_6	$I_3(0.67)$	-	-

(δ)

Σχήμα 2.3: (α) *Πίνακας ομοιότητας χρηστών*, (β) Οι *πλησιέστεροι γείτονες των υπό εξέταση χρηστών* σε φθίνουσα κατάταξη βάσει της ομοιότητας, (γ) *Πίνακας ομοιότητας στοιχείων*, (δ) Οι *πλησιέστεροι γείτονες των υπό εξέταση στοιχείων* σε φθίνουσα κατάταξη βάσει της ομοιότητας.

2.3.2 Παράγοντες Δευτέρου Σταδίου

Μέγεθος της λίστας συστάσεων. Το μέγεθος N της λίστας προτεινόμενων στοιχείων αντιστοιχεί σε μία *αντιστάθμιση (trade off)* μεταξύ του μέτρου *ακριβείας* της παραπάνω λίστας και του μέτρου *ανάκλησης*: όσο δηλαδή αυξάνεται ο αριθμός N των προτεινόμενων προϊόντων, τόσο αναμένεται να μειωθεί και η *ακρίβεια* του αλγορίθμου συστάσεων - η οποία και θα οριστεί παρακάτω. Σε σχετικές εργασίες ([Karypis, 2001], [Sarwar et al., 2001]) ο αριθμός N των προτεινόμενων στοιχείων συνήθως λαμβάνει τιμές από 5 ως 20.

Κατώφλι θετικής βαθμολογίας. Είναι προφανές ότι οι συστάσεις στοιχείων πρέπει να είναι "θετικές". Δεν αποτελεί επιτυχία η σύσταση ενός στοιχείου που θα βαθμολογηθεί τελικά από τον χρήστη με βαθμό 1 στην κλίμακα βαθμολογίας 1-5. Με άλλα λόγια, "αρνητικά" βαθμολογημένα

στοιχεία δεν θα έπρεπε να συμβάλλουν στην αύξηση του ποσοστού ακριβείας, και για τον λόγο αυτόν χρησιμοποιούμε ένα *κατώφλι θετικής βαθμολογίας* P_τ (*Positive Rating Threshold*) για τη σύσταση στοιχείων των οποίων η βαθμολογία δεν είναι μικρότερη από αυτήν την τιμή. Εάν δεν χρησιμοποιηθεί μια τιμή P_τ, τότε τα αποτελέσματα των συστάσεων μπορεί να αποβούν παραπλανητικά δεδομένου ότι οι αρνητικές βαθμολογίες είναι δυνατόν να επηρεάσουν αρνητικά την αποτελεσματικότητα των *αλγορίθμων ΣΦ*.

Δημιουργία λίστας συστάσεων. Η συχνότερα χρησιμοποιούμενη τεχνική για τη δημιουργία μιας λίστας συστάσεων N στοιχείων είναι εκείνη που μετρά τη *συχνότητα εμφάνισης κάθε στοιχείου* εντός της γειτονιάς του υπό εξέταση χρήστη και ακολούθως προτείνει τα *συχνότερα εμφανιζόμενα N στοιχεία* [Sarwar et al., 2000a]. Στο εξής, αυτή η τεχνική θα αναφέρεται ως *Σύσταση του πιο συχνού στοιχείου στη γειτονιά* (*Most-Frequent Item in Neighborhood - MFiN*). Η *τεχνική MFiN* είναι δυνατόν να εφαρμοσθεί και στους δύο αλγορίθμους *ΣΦ* (δηλαδή *ΣΦ βάσει χρήστη* και *ΣΦ βάσει στοιχείου*). Στο Σχήμα 2.4 παρουσιάζουμε τον ψευδοκώδικα με τις δύο κύριες μεθόδους που έχουν προταθεί στη βιβλιογραφία για τη δημιουργία της *λίστας σύστασης των κορυφαίων-N στοιχείων βάσει του πιο συχνού στοιχείου μέσα στη γειτονιά (MFiN)* για τον *User-based* και τον *Item-based* αλγόριθμο αντίστοιχα.

Ο Karypis [Karypis, 2001], ωστόσο, πρότεινε για την παραγωγή της λίστας προτεινόμενων στοιχείων μια άλλη τεχνική που εξετάζει επιπρόσθετα τον βαθμό ομοιότητας των χρηστών ή στοιχείων που βρίσκονται στη γειτονιά του υπό εξέταση χρήστη με τον ίδιο. Πιο συγκεκριμένα, αυτή η τεχνική λαμβάνει υπόψη το ότι οι ομοιότητες των k *γειτόνων* με τον υπό εξέταση χρήστη μπορεί να ποικίλλουν σημαντικά. Έτσι, για κάθε στοιχείο στη γειτονιά του υπό εξέταση χρήστη η τεχνική αυτή προσμετρά εξίσου όχι μόνο τον αριθμό εμφανίσεων του κάθε στοιχείου αλλά και την ομοιότητα του κάθε γειτονικού χρήστη που το βαθμολόγησε με τον υπό εξέταση χρήστη. Τα N στοιχεία με το υψηλότερο άθροισμα ομοιοτήτων είναι και αυτά που τελικά προτείνονται. Αυτή λοιπόν η τεχνική θα αναφέρεται στη συνέχεια ως *Σύσταση του στοιχείου με το Υψηλότερο Άθροισμα Ομοιοτήτων* (*Highest Sum of Similarities - HSS*). Να σημειωθεί ότι η *τεχνική HSS* είναι εφαρμόσιμη μόνο στον *αλγόριθμο ΣΦ βάσει στοιχείου*.

Ως μια επέκταση των προαναφερθέντων δύο κριτηρίων (*MFiN* και *HSS*) θα μπορούσε κανείς απλά να χρησιμοποιήσει τις προβλεφθείσες για κάθε στοιχείο βαθμολογίες, και στη συνέχεια να κατατάξει τα στοιχεία βάσει των τιμών βαθμολογίας που έχουμε προβλέψει για αυτά. Αναλυτικότερα, η προβλεφθείσα βαθμολογία [McLauglin and Herlocker, 2004] ενός χρήστη u για ένα στοιχείο i υπολογίζεται από τις Εξισώσεις 2.3 και 2.4 για τις περιπτώσεις του *αλγορίθμου ΣΦ βάσει χρήστη* και του *αλγορίθμου ΣΦ βάσει στοιχείου* αντίστοιχα:

$$p_{u,i} = \overline{r}_u + \frac{\sum_{v \in U} \mathrm{sim}(u,v)(r_{v,i} - \overline{r}_v)}{\sum_{v \in U} |\mathrm{sim}(u,v)|} \tag{2.3}$$

$$p_{u,i} = \overline{r}_i + \frac{\sum_{j \in I} \mathrm{sim}(i,j)(r_{u,j} - \overline{r}_j)}{\sum_{j \in I} |\mathrm{sim}(i,j)|} \tag{2.4}$$

```
Vector UBCF-MFiN(u, NN(u), I, N)
//User u
//Set NN(u) (neighbors)
//int I (set of items)
//int N (top-N items)
//Pτ (threshold for positive rating)
begin
    for i=1 to I
        f[i] = 0;
    for each v ∈ NN(u)
        for i = 1 to I
            if r[v][i] ≥ Pτ
                f[i] = f[i] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

(α)

```
Vector IBCF-MFiN(u, NN(i), I, N)
//User u
//Set NN(i) (neighbors)
//int I (set of items)
//int N (top-N items)
//Pτ (threshold for positive rating)
begin
    for j = 1 to I
        f[j] = 0;
    for j = 1 to I
        if r[u][j] ≥ Pτ
            for each v ∈ NN(i)
                f[v] = f[v] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

(β)

Σχήμα 2.4: Δημιουργία της *λίστας των κορυφαίων-N στοιχείων* με βάση τον *αλγόριθμο μέτρησης των συχνότερα εμφανιζόμενων στοιχείων μέσα στη γειτονιά (MFiN)* για τους εξής: (α) αλγόριθμο *ΣΦ βάσει χρήστη* και (β) αλγόριθμο *ΣΦ βάσει στοιχείου.*

.

Επομένως, κατατάσσουμε με φθίνουσα σειρά τα στοιχεία σύμφωνα με την τιμή βαθμολογίας που προβλέπουμε για αυτά, και στη συνέχεια προτείνουμε στον υπό εξέταση χρήστη τα κορυφαία εξ αυτών N στοιχεία.[1] Αυτή η μέθοδος κατάταξης των στοιχείων θα αναφέρεται στο εξής ως *Σύσταση του στοιχείου με την Υψηλότερη Προβλεφθείσα Βαθμολογία* (*Higher Predicted Rating - HPR*). Η *HPR* προκρίνει για σύσταση τα στοιχεία εκείνα που είναι πιθανό να λάβουν υψηλή βαθμολογία από τον υπό εξέταση χρήστη. Παρόλα αυτά θα πρέπει να τονίσουμε ότι η *HPR* εμφανίζει μικρή αποτελεσματικότητα στη σύσταση των *κορυφαίων-N στοιχείων.*

Μια άλλη, τώρα, μέθοδος κατάταξης των στοιχείων αθροίζει τις θετικές βαθμολογίες των στοιχείων της γειτονιάς αντί απλώς να μετρά τη συχνότητά τους. Αυτό το κριτήριο δηλώνεται ως μέτρο *Υψηλότερου Αθροίσματος Βαθμολογίας* (*Higher Sum Rating - HSR*). Η λίστα συστάσεων αποτελείται από τα κορυφαία N στοιχεία που έχουν το υψηλότερο άθροισμα. Η λογική του *HSR* είναι ότι λαμ-

[1] Εάν λιγότερα από N στοιχεία έχουν θετικές βαθμολογίες (π.χ. με προβλεπόμενη βαθμολογία μικρότερη από το P_τ), τότε λιγότερα από N στοιχεία παραμένουν στη λίστα των προτεινόμενων στοιχείων.

βάνει υπόψη τόσο τη συχνότητα εμφάνισης ενός στοιχείου (όπως το *MFiN*) όσο και το ύψος της βαθμολογίας, επειδή στοχεύει στο να ευνοεί στοιχεία που εμφανίζονται συχνότερα στη γειτονιά *και* έχουν πάρει και τις υψηλότερες βαθμολογίες. Έστω για παράδειγμα ένα στοιχείο i που έχει οριακά μικρότερη συχνότητα εμφάνισης από ένα j. Εάν λοιπόν το i παίρνει πολύ υψηλότερες βαθμολογίες από το j, τότε η *HPR* θα το προτάξει έναντι του i, ενώ το *MFiN* θα προτάξει το j έναντι του i.

Οι αλγόριθμοι παραγωγής της λίστας προτεινόμενων στοιχείων συνοψίζονται στον Πίνακα 2.3.

Όνομασία Αλγορίθμου	**Αγγλική Ονομασία**
Σύσταση του πιο *συχνού στοιχείου στη γειτονιά*	*Most-Frequent Item in Neighborhood - MFiN*
Σύσταση του στοιχείου με το *Υψηλότερο Άθροισμα Ομοιοτήτων*	*Highest Sum of Similarities - HSS*
Σύσταση του στοιχείου με την *Υψηλότερη Προβλεφθείσα Βαθμολογία*	*Higher Predicted Rating - HPR*
Σύσταση του στοιχείου με το *Υψηλότερο Άθροισμα Βαθμολογίας*	*Higher Sum Rating - HSR*

Πίνακας 2.3: Αλγόριθμοι για την παραγωγή της λίστας προτεινόμενων στοιχείων.

2.3.3 Παράγοντες Τρίτου Σταδίου

Ορισμός μιας μεθόδου ως αλγορίθμου αναφοράς (baseline algorithm). Πολλές από τις υπάρχουσες πειραματικές αποτιμήσεις της *αποτελεσματικότητας* της λίστας προτεινόμενων στοιχείων στερούνται τη δυνατότητα σύγκρισης μ' έναν απλό αλγόριθμο που θα χρησιμοποιηθεί ως σημείο αναφοράς (*baseline algorithm*). Ένας τέτοιος αλγόριθμος οφείλει να είναι απλός και να υποδεικνύει τι είναι δυνατόν να επιτευχθεί με τη λιγότερη δυνατή προσπάθεια. Έτσι θα μπορούσαμε να δούμε και την πραγματική βελτίωση που μπορεί να επιφέρει ένας *αλγόριθμος ΣΦ*. Στη βιβλιογραφία των συστημάτων συστάσεων συχνά χρησιμοποιείται ως *αλγόριθμος αναφοράς* αυτός που απλώς προτείνει τα κορυφαία-N στοιχεία που είναι τα συχνότερα θετικώς βαθμολογημένα σ' ολόκληρο το σύνολο των δεδομένων εκπαίδευσης. Αυτός ο αλγόριθμος δηλώνεται ως *Most Popular Items Recommendation* ή, εν συντομία, *POP*. Ο *POP* είναι πολύ απλός και, όπως θα δείξουμε με πειραματικά δεδομένα, ιδιαίτερα αποτελεσματικός. Αναλυτικότερα, δοκιμές με τον *POP* στο σύνολο των δεδομένων *Movielens-100K* δείχνουν ότι, όταν απλώς προτείνουμε τις πέντε υψηλότερα βαθμολογημένες (δημοφιλέστερες) ταινίες, τότε η ακρίβεια αγγίζει το 22%. Επομένως, τα στοιχεία που συχνά παίρνουν υψηλή βαθμολογία είναι πιθανότερο να επιλεγούν από την πλειονότητα των χρηστών. Για τους παραπάνω λόγους, ο *POP* κρίνεται ως ένας καλός *αλγόριθμος αναφοράς* και αξιολόγησης της πραγματικής ποιότητας των υπαρχόντων *αλγορίθμων ΣΦ*.

Παρελθοντικά/μελλοντικά δεδομένα. Στις πραγματικές, μη πειραματικές, εφαρμογές οι συστάσεις βασίζονται μόνο στις τρέχουσες διαθέσιμες βαθμολογίες του υπό εξέταση χρήστη. Στις περισσότερες από τις σχετικές με το θέμα εργασίες ωστόσο ολόκληρο το σύνολο των στοιχείων που έχουν βαθμολογηθεί θεωρείται *εκ προοιμίου* γνωστό. Άρα, για μια ρεαλιστικότερη αξιολόγηση οι συστάσεις θα πρέπει να εξετάζουν τη διαίρεση των στοιχείων σε δύο σύνολα: (i) στα παρελθόντα στοιχεία και (ii) στα μελλοντικά στοιχεία. Για να το καταστήσουμε σαφές, ας εξετάσουμε ένα

παράδειγμα: Έστω ότι ο υπό εξέταση χρήστης έχει βαθμολογήσει 10 στοιχεία. Είναι ρεαλιστικό να μετρήσουμε πώς ανταποκρίνονται οι αλγόριθμοι όταν χρησιμοποιούμε έναν μικρότερο αριθμό στοιχείων από αυτά που έχει βαθμολογήσει. Δηλαδή, εάν χρησιμοποιήσουμε 2 στοιχεία (παρελθόντα) από τα 10, μπορούμε να ανακαλύψουμε τα υπόλοιπα 8 (μελλοντικά). Εάν μόνο ένα κλάσμα στοιχείων περιλαμβάνεται στο σύνολο των παρελθόντων, τότε η *ακρίβεια* αναμένεται να μειωθεί συγκρινόμενη με την καλύτερη περίπτωση.

Μετρικές Αξιολόγησης. Στα συστήματα συστάσεων υπάρχουν δύο σενάρια πρόβλεψης:

1. *Πρόβλεψη βαθμολογίας στοιχείου* (*Rating Prediction Task*), π.χ. η βαθμολογία που θα πάρει ένα στοιχείο στην κλίμακα 1 έως 5, και

2. *Πρόβλεψη λίστας προτεινόμενων στοιχείων* (*Top-N Item Recommendation Task*), π.χ. η παραγωγή μιας λίστας με τα *κορυφαία-N προτεινόμενα στοιχεία.*

Για έναν υπό εξέταση χρήστη που λαμβάνει μια *λίστα N-κορυφαίων προτεινόμενων στοιχείων*, έστω ότι το R δηλώνει τον αριθμό των *συναφών προτεινόμενων στοιχείων* (τα στοιχεία της λίστας συστάσεων που βαθμολογήθηκαν υψηλότερα από P_τ από τον υπό εξέταση χρήστη). Ορίζουμε τα εξής:

- *Ακρίβεια (Precision)* είναι ο λόγος του R στο N.

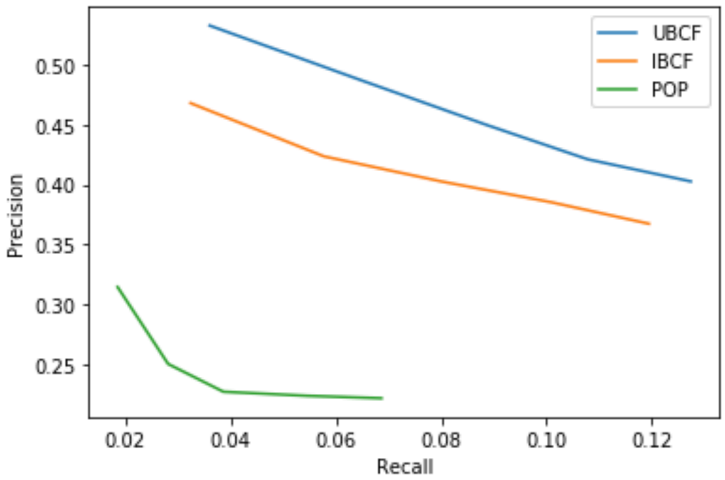
- *Ανάκληση (Recall)* είναι ο λόγος του R στον συνολικό αριθμό των *συναφών στοιχείων* με τον υπό εξέταση χρήστη (όλα τα στοιχεία που βαθμολογήθηκαν από αυτόν υψηλότερα από το *κατώφλι θετικής βαθμολογίας* P_τ).

2.4 Συγκριτική Μελέτη Επίδοσης Αλγορίθμων ΣΦ

Στην ενότητα αυτή μελετούμε την επίδοση των *αλγορίθμων ΣΦ*. Ελέγχονται αμφότεροι οι *αλγόριθμοι ΣΦ*, *βάσει χρήστη* και *βάσει στοιχείου*, καθώς και ο *αλγόριθμος αναφοράς POP* που προτείνει τα πιο δημοφιλή στοιχεία. Οι πρόσθετοι παράγοντες που μεταχειριζόμαστε ως παραμέτρους είναι οι εξής: το μέγεθος γειτονιάς (k, προεπιλεγμένη τιμή 5), το μέγεθος της λίστας προτεινόμενων στοιχείων (N, προεπιλεγμένη τιμή 5), το μέγεθος του συνόλου δεδομένων εκπαίδευσης (προεπιλεγμένη τιμή 80%). Η αξιολόγηση επιτελείται με τις μετρικές της *ακρίβειας* και της *ανάκλησης*.

Τα αποτελέσματα των *Precision-Recall καμπυλών* απεικονίζονται στο Σχήμα 2.5. Τα αποτελέσματα αυτά καταδεικνύουν ότι η *UBCF* είναι αποτελεσματικότερη από την *IBCF*. Η διαφορά στην *ακρίβεια* είναι σχεδόν 5%, ενώ αναφορικά με την *ανάκληση* είναι 1% όπως φαίνεται από τον Πίνακα 2.4. Ο λόγος είναι ότι ο αλγόριθμος *UBCF* εστιάζει περισσότερο στις προτιμήσεις του υπό εξέταση χρήστη. Αντίθετα, ο αλγόριθμος *IBCF* πιθανόν να μην αντανακλά άμεσα τις προτιμήσεις του υπό εξέταση χρήστη, αλλά κυρίως τις ομοιότητες μεταξύ των στοιχείων. Τέλος, πρέπει

να παρατηρήσουμε ότι ο αλγόριθμος *IBCF* δεν επιβαρύνεται χρονικά με τον υπολογισμό του *πίνακα ομοιότητας στοιχείων* (υπολογίζεται *offline*), γεγονός που του δίνει πλεονέκτημα σε σχέση με τον αλγόριθμο *UBCF* ως προς τον χρόνο εκτέλεσης. Ακόμη, η δημιουργία της *λίστας των κορυφαίων N προτεινόμενων στοιχείων* στον *UBCF* επιβαρύνει επιπλέον τη *διεργασία ΣΦ*. Ο λόγος είναι ότι ο αλγόριθμος *UBCF* υπολογίζει πρώτα τον *πίνακα ομοιότητας* των χρηστών και κατόπιν μετρά τη *συχνότητα εμφάνισης* των στοιχείων μέσα στον *πίνακα ομοιότητας* μεταξύ των χρηστών. Σε αντίθεση λοιπόν με τον αλγόριθμο *IBCF*, όλη η *διεργασία ΣΦ* ολοκληρώνεται μέσα σ' ένα βήμα, απευθείας στον *πίνακα ομοιότητας στοιχείων*.



Σχήμα 2.5: *Καμπύλη Precision-Recall* για τις μεθόδους *UBCF*, *IBCF*, και *POP*.

.

Αλγόριθμος	Σύνολο δεδομένων *Movielens 100K*	
	Ακρίβεια	*Ανάκληση*
UBCF	0.4026	0.1274
IBCF	0.3673	0.1194
POP	0.2212	0.0686

Πίνακας 2.4: Μέση τιμή *ακρίβειας* και *ανάκλησης* όταν προτείνουμε λίστα με τις 5 κορυφαίες ταινίες στο *Movielens 100K dataset*.

2.5 Συμπεράσματα

Στο κεφάλαιο αυτό μελετήσαμε διεξοδικά τους *αλγόριθμους ΣΦ βάσει των κοντινότερων γειτόνων* (*χρηστών* ή *στοιχείων*), αναδεικνύοντας διάφορους σημαντικούς παράγοντες και προκλήσεις που επηρεάζουν το *ΣΦ* και, ειδικότερα, θέματα που αφορούν: (i) στη *διαμόρφωση της γειτονιάς χρήστη*,

(ii) στη *δημιουργία της λίστας των κορυφαίων N προτεινόμενων στοιχείων*, και (iii) στην αποτίμηση της *ποιότητας της λίστας προτεινόμενων στοιχείων*. Συνοπτικά, καταλήγουμε στα εξής συμπεράσματα:

- Οι αλγόριθμοι για την *παραγωγή της λίστας των κορυφαίων N προτεινόμενων στοιχείων* είναι ένα σημαντικό πρόβλημα που χρήζει περαιτέρω διερεύνησης. Το κριτήριο *HSR* θα μπορούσε να αντικαταστήσει το *MFiN* ως περισσότερο αποτελεσματικό.

- Τα αποτελέσματα σχετικών εργασιών καταδεικνύουν ότι ακολουθώντας τις καλύτερες επιλογές για τους αλγορίθμους *UBCF* και *IBCF*, ο πρώτος υπερέχει έναντι του δεύτερου.

- Ο αλγόριθμος *Most Popular Items Recommendation* είναι χρήσιμος ως μέτρο σύγκρισης και σημείο αναφοράς για τον ορθότερο χαρακτηρισμό της επίδοσης των αλγορίθμων *ΣΦ*.

Τέλος, πρέπει να παρατηρήσουμε ότι ο *αλγόριθμος IBCF* υπερέχει έναντι του *αλγορίθμου UBCF* ως προς τον χρόνο εκτέλεσης και παραγωγής της λίστας προτεινόμενων στοιχείων.

2.6 Ερωτήσεις Κεφαλαίου

1. Πώς λειτουργεί η τεχνική του *συνεργατικού φιλτραρίσματος*;

2. Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα του *συνεργατικού φιλτραρίσματος*;

3. Ποιες μέθοδοι χρησιμοποιούνται για τον υπολογισμό της *ομοιότητας των χρηστών*; Θα μπορούσατε να φανταστείτε άλλα *μέτρα ομοιότητας*;

4. Ποιες μέθοδοι χρησιμοποιούνται για τον υπολογισμό της *ομοιότητας των στοιχείων*;

5. Πώς λειτουργεί ο *αλγόριθμος συνεργατικού φιλτραρίσματος βάσει χρήστη (UBCF)*;

6. Πώς μπορεί η μέθοδος *συνεργατικού φιλτραρίσματος* να λάβει υπόψη το γεγονός ότι μια βαθμολογία είναι παλιά και μπορεί να μην είναι πλέον σχετική;

7. Για έναν χρήστη που εγγράφεται για πρώτη φορά στην υπηρεσία μας: ποιος είναι ο καλύτερος τρόπος να επιλέξουμε τα στοιχεία που θέλουμε να αξιολογήσει ο χρήστης; Να αναφέρετε μια καλή μέθοδο επιλογής στοιχείων.

8. Πώς λειτουργεί ο *αλγόριθμος συνεργατικού φιλτραρίσματος βάσει στοιχείου (IBCF)*;

9. Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα του *αλγορίθμου συνεργατικού φιλτραρίσματος βάσει στοιχείου (IBCF)*;

2.7 Ασκήσεις Προγραμματισμού

Άσκηση 1: Κατασκευή ενός User-User Similarity Matrix στη γλώσσα Python

Ένας εύκολος τρόπος για να προβλέψουμε με επιτυχία μια λίστα με προτεινόμενες ταινίες για έναν υπό εξέταση θεατή είναι να λάβουμε υπόψη τα είδη ταινιών (*movie genres*) που τον ενδιαφέρουν, και να προσπαθήσουμε να αναζητήσουμε άλλους θεατές που να είναι *παρόμοιοι* με αυτόν. Στη συνέχεια να του προτείνουμε τα είδη ταινιών (δραματικές, ρομαντικές κ.λπ.) που ενδιαφέρουν τους *παρόμοιους* μ' αυτόν θεατές. Για να το κάνουμε αυτό, θα χρειαστούμε έναν τρόπο για να μετρήσουμε πόσο όμοιοι είναι δύο θεατές σε σχέση με τις προτιμήσεις τους ως προς τα είδη ταινιών. Για τον σκοπό αυτόν θα χρησιμοποιήσουμε την *ομοιότητα συνημιτόνου*, ώστε να μετρήσουμε πόσο όμοια είναι τα διανύσματα μεταξύ δύο προφίλ θεατών. Στη συνέχεια, θα εφαρμόσουμε τη μέθοδο αυτή σε διανύσματα που εμπεριέχουν είτε 0 (μηδενικά) είτε 1 (μονάδες), όπου κάθε διάνυσμα **v** αντιπροσωπεύει τις προτιμήσεις ενός θεατή για διαφορετικά είδη ταινιών (κωμωδίες, πολεμικές κ.λπ.). Το **v[i]** θα είναι 1 εάν στον θεατή αρέσει το συγκεκριμένο είδος ταινίας *i*, και 0 εάν όχι.

Μας δίνεται λοιπόν μια δομή δεδομένων μιας λίστας με "λεξικά" (*list with dictionaries*) με τους παρακάτω 6 θεατές υπό το αναγνωριστικό (id) τους:

```
import pandas as pd
import numpy as np
import math
from typing import Dict, List, Tuple
Vector = List[float]
from sklearn.metrics.pairwise import cosine_similarity
from collections import defaultdict
import copy

viewers = [
    { "viewer_id": 0 },
    { "viewer_id": 1 },
    { "viewer_id": 2 },
    { "viewer_id": 3 },
    { "viewer_id": 4 },
    { "viewer_id": 5 }
]
```

Επίσης, μας δίνεται μια δομή δεδομένων "λεξικού" (*dictionary*) με τα ενδιαφέροντά τους ως προς τα είδη ταινιών:

```
viewer_genres = [
(0, "Action"), (0, "Adventure"), (0, "Animation"), (0, "Biography"), (0, "Black Comedy"), (0, "Comedy"), (0, "Concert"),
(1, "Crime"), (1, "Documentary"), (1, "Concert"), (1, "Animation"), (1, "Drama"),
(2, "Educational"), (2, "Family"), (2, "Fantasy"), (2, "Film Noir"), (2, "Game-Show"), (2, "History"),
(3, "Horror"), (3, "Educational"), (3, "Music"), (3, "Musical"), (3, "Mystery"),
(4, "Mystery"), (4, "Musical"), (4, "Reality-TV"), (4, "Romance"),
(5, "Educational"), (5, "Horror"), (5, "Biography"), (5, "Romantic Comedy"), (5, "Sci-Fi"), (5, "Short Film")
]
```

Για να διαχειριστούμε τα ενδιαφέροντα των θεατών πιο αποτελεσματικά, μπορούμε να καταργήσουμε τα αναγνωριστικά (*ids*) τους και να δημιουργήσουμε μια δομή δεδομένων μίας *λίστας που εμπεριέχεται σε λίστα* (*list within list*) όπου θα παραμείνουν μόνο τα ενδιαφέροντα των θεατών. Τονίζεται ότι κάθε επιμέρους υπο-λίστα αυτής της λίστας περιλαμβάνει τις προτιμήσεις ενός θεατή. Επίσης, ένα είδος ταινίας μπορεί να εμφανίζεται σε περισσότερες της μιας υπο-λίστες. Τη δομή δεδομένων *μίας λίστας που εμπεριέχεται σε λίστα* (*list within list*) που δημιουργήσαμε την ονομάζουμε *movie_genres*.

```
movie_genres = [
 ["Action", "Adventure", "Animation", "Biography", "Black Comedy", "Comedy", "Concert"],
 ["Crime", "Documentary", "Concert", "Animation", "Drama"],
 ["Educational", "Family", "Fantasy", "Film Noir", "Game-Show", "History"],
 ["Horror", "Educational", "Music", "Musical", "Mystery"],
 ["Mystery", "Musical", "Reality-TV", "Romance"],
 ["Educational", "Horror", "Biography", "Romantic Comedy", "Sci-Fi", "Short Film"]]
```

Στη συνέχεια αφαιρούμε τις διπλότυπες εμφανίσεις ενός είδους ταινιών από τη λίστα *movie_genres* με τον παρακάτω κώδικα:

```
def unique(list1):

    # initialize a null list
    unique_list = []

    for i in range(len(list1)):
        for j in range(len(list1[i])):
                # check if exists in unique_list or not
                 if list1[i][j] not in unique_list:
                  unique_list.append(list1[i][j])
    return unique_list

print("The unique movie genres are:")
unique(movie_genres)
```

Έτσι προκύπτει μια λίστα με τα 25 είδη ταινιών που εμφανίζονται μία φορά:

The unique movie genres are: ['Action', 'Adventure', 'Animation', 'Biography', 'Black Comedy', 'Comedy', 'Concert', 'Crime', 'Documentary', 'Drama', 'Educational', 'Family', 'Fantasy', 'Film Noir', 'Game-Show', 'History', 'Horror', 'Music', 'Musical', 'Mystery', 'Reality-TV', 'Romance', 'Romantic Comedy', 'Sci-Fi', 'Short Film']

Στη συνέχεια χρειαζόμαστε ένα διάνυσμα που θα εμπεριέχει τις προτιμήσεις (με 1 ή 0) του κάθε θεατή ως προς τα 25 διαφορετικά είδη ταινιών. Για να το πετύχουμε αυτό πρέπει απλώς να επαναδημιουργήσουμε τη λίστα όπου το κάθε είδος ταινίας θα εμφανίζεται μόνο μία φορά, και κατόπιν θα τοποθετούμε το 1 εάν ο θεατής δείχνει ενδιαφέρον για ένα συγκεκριμένο είδος ταινίας (π.χ. πολεμικές), και το 0 εάν όχι, με τη χρήση του παρακάτω κώδικα:

```
viewer_genre_vectors = []
for viewer_genres in movie_genres:
    viewer_genre_vectors.append ([0] * 25)
# print(user_genre_vectors)
for viewer_genres in movie_genres:
    for genre in unique_genres:
        if genre in viewer_genres:
```

```
            viewer_genre_vectors[movie_genres.index(viewer_genres)][unique_genres.index(genre)] = 1

print("The vectors that represent the 6 viewer profiles are:")
viewer_genre_vectors
```

The vectors that represent the 6 viewer profiles are:

[[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1]]

Επομένως το *viewer_genre_vectors*[i][j] ισούται με 1 εάν ο χρήστης i ενδιαφέρεται για το είδος ταινιών j, και με 0 εάν όχι.

Δεδομένου ότι έχουμε ένα μικρό σύνολο δεδομένων, ο υπολογισμός της *ομοιότητας συνημιτόνου* μεταξύ των 6 χρηστών γίνεται εύκολα, και ο κώδικας *Python* που τον υλοποιεί δίνεται παρακάτω όπου γίνεται χρήση της έτοιμης συνάρτησης από το *sklearn library* (στην αρχή του κώδικα είχαμε κάνει *import* τη *library* με τη χρήση της εντολής *from sklearn.metrics.pairwise import cosine_similarity)*:

```
def viewer_viewer_cosine_similarity(train):
    similarities = cosine_similarity(train)
    return similarities

viewer_sim = viewer_viewer_cosine_similarity(viewer_genre_vectors)

print("The viewer-viewer similarity matrix is the following: ")
print(pd.DataFrame(viewer_sim).round(3))
```

```
The viewer-viewer similarity matrix is the following :
        0      1      2      3      4      5
0   1.000  0.338  0.000  0.000  0.000  0.154
1   0.338  1.000  0.000  0.000  0.000  0.000
2   0.000  0.000  1.000  0.183  0.000  0.167
3   0.000  0.000  0.183  1.000  0.447  0.365
4   0.000  0.000  0.000  0.447  1.000  0.000
5   0.154  0.000  0.167  0.365  0.000  1.000
```

Συνεπώς η συνάρτηση *viewer_sim*[i][j] μας επιστρέφει την *ομοιότητα* μεταξύ των θεατών i και j. Για παράδειγμα, ο θεατής 0 και ο θεατής 5 έχουν ως μόνο κοινό τους στοιχείο το ενδιαφέρον τους για το είδος ταινιών *Biography* και έχουν *ομοιότητα* ίση με *viewer_sim*[0][5]= 0.154. Επίσης, τον ίδιο *viewer-viewer similarity matrix* μπορούμε να παράξουμε χωρίς τη χρήση της έτοιμης συνάρτησης *cosine_similarity* από τη βιβλιοθήκη *sklearn*. Πιο συγκεκριμένα, με τη χρήση του παρακάτω κώδικα μπορούμε να χρησιμοποιήσουμε τις *συναρτήσεις dot* (*εσωτερικό γινόμενο διανυσμάτων*) και *cosine_similarity (ομοιότητα συνημιτόνου)*,

$$sim(\vec{u},\vec{v}) = cos(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{||\vec{u}||\cdot||\vec{v}||}, \tag{2.5}$$

για να λάβουμε το ίδιο ακριβώς αποτέλεσμα:

```
def dot(v: Vector, w: Vector) -> float:
    return sum(v_i * w_i for v_i, w_i in zip(v, w))

def cosine_similarity(v1: Vector, v2: Vector) -> float:
    return dot(v1, v2) / math.sqrt(dot(v1, v1) * dot(v2, v2))

viewer_sim = []
for viewer_genres in movie_genres:
    viewer_sim.append ([0] * 6)
i=0
for vector_i in viewer_genre_vectors:
    j=0
    for vector_j in viewer_genre_vectors:
        viewer_sim[i][j] = cosine_similarity(vector_i, vector_j
    )

        j=j+1
    i=i+1
```

```python

print("The viewer-viewer similarity matrix is the following: ")
print(pd.DataFrame(viewer_sim).round(3))
```

```
The viewer-viewer similarity matrix is the following :
       0      1      2      3      4      5
0  1.000  0.338  0.000  0.000  0.000  0.154
1  0.338  1.000  0.000  0.000  0.000  0.000
2  0.000  0.000  1.000  0.183  0.000  0.167
3  0.000  0.000  0.183  1.000  0.447  0.365
4  0.000  0.000  0.000  0.447  1.000  0.000
5  0.154  0.000  0.167  0.365  0.000  1.000
```

Τέλος, τον ίδιο *viewer-viewer similarity matrix* μπορούμε να παράξουμε με απλές πράξεις πινάκων της γραμμικής άλγεβρας. Έστω, λοιπόν, ότι αναπαριστούμε τον *πίνακα προτιμήσεων θεατών-ειδών ταινιών* ($viewer_genre_vectors[i][j]$) ως R όπου κάθε γραμμή (του R) αντιστοιχεί σ' έναν θεατή και κάθε στήλη αντιστοιχεί σ' ένα είδος ταινιών.
Από τον πίνακα R (ο οποίος μπορεί να θεωρηθεί και ένας *διμερής γράφος* μεταξύ θεατών και ειδών ταινιών) μπορούμε να ορίσουμε τον πίνακα $P, m \times m$ ως έναν διαγώνιο πίνακα του οποίου το i-οστό διαγώνιο στοιχείο είναι ο βαθμός του κόμβου-θεατή i (degree of node i) , ο οποίος και καταγράφει τον συνολικό αριθμό των ειδών ταινιών που αρέσουν στον θεατή i.
Γνωρίζουμε, ακόμη, ότι το διάνυσμα ενός θεατή ορίζεται από μία γραμμή του R. Επιπλέον, η *frobenius norm (νόρμα) του διανύσματος του θεατή* i ορίζεται από τη σχέση $\sqrt{\sum_{k=1}^{m} R_{ik}^2}$. Το άθροισμα είναι στην πραγματικότητα ο αριθμός των διαφορετικών ειδών ταινιών που αρέσουν σ' αυτόν τον θεατή (καθώς το R_{ik} μπορεί να είναι είτε 0 είτε 1, έχουμε $R_{ik} = R_{ik}^2$). Έτσι, η *νόρμα του διανύσματος του θεατή* i είναι στην πραγματικότητα $\sqrt{P_{ii}}$. (Σημείωση: Για να υπολογίσουμε την τετραγωνική ρίζα ενός πίνακα για κάθε επιμέρους στοιχείο του, μπορούμε να το πράξουμε ως πίνακα στη δύναμη του $\frac{1}{2}$). Επομένως έχουμε:

$$\text{cos-sim}\Big(\text{ viewer }_i, \text{ viewer }_j\Big) = \frac{R_{i.} \cdot R_{j.}}{\sqrt{P_{ii}}\sqrt{P_{jj}}} \quad = \frac{\sum_{k=1}^{m} R_{ik} R_{jk}}{\sqrt{P_{ii}}\sqrt{P_{jj}}}$$

Ο πίνακας P είναι διαγώνιος, και οι διαγώνιοι συντελεστές του είναι όλοι μη μηδενικοί, οπότε μπορούμε να συμβολίσουμε με $P^\star$ τον διαγώνιο πίνακα του οποίου τα διαγώνια στοιχεία ορίζονται από τη σχέση $P_{ii}^\star = P_{ii}^{-1/2}$. Έτσι, ο τελικός *viewer-viewer similarity* S_{vv} *matrix* μπορεί να εκφραστεί ως προς P και R :

$$S_{vv} = P^\star \times R \times R^T \times P^\star$$

```python
from scipy.linalg import fractional_matrix_power

R = viewer_genre_vectors.copy()
P = np.diag(np.sum(R, axis=1))
P = fractional_matrix_power(P, -0.5)
PR =  np.matmul(P, R)

#viewer-viewer similarity matrix
Svv=np.matmul(PR,np.transpose(PR));
print(pd.DataFrame(Svv).round(3))
```

```
The viewer-viewer similarity matrix is the following :
       0      1      2      3      4      5
0  1.000  0.338  0.000  0.000  0.000  0.154
1  0.338  1.000  0.000  0.000  0.000  0.000
2  0.000  0.000  1.000  0.183  0.000  0.167
3  0.000  0.000  0.183  1.000  0.447  0.365
4  0.000  0.000  0.000  0.447  1.000  0.000
5  0.154  0.000  0.167  0.365  0.000  1.000
```

Άσκηση 2: Χρησιμοποιώντας ως βάση τον παραπάνω κώδικα δημιουργήστε ένα *Notebook Jupyter* για την υλοποίηση του *item-item similarity matrix* μαζί με διευκρινιστικά σχόλια με την περιγραφή της αλγοριθμικής διαδικασίας που προκύπτει από κάθε κελί κώδικα.

Βιβλιογραφία

D. Goldberg, D. Nichols, M. Brian, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61--70, 1992.

K. Goldberg, T. Roeder, T. Gupta, and C. Perkins. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133--151, 2001.

J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings ACM SIGIR Conference*, pages 230--237, 1999.

J. Herlocker, J. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287--310, 2002.

G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings ACM CIKM Conference*, pages 247--254, 2001.

R. McLauglin and J. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings ACM SIGIR Conference*, pages 329--336, 2004.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering on netnews. In *Proccedings Conference Computer Supported Collaborative Work*, 1994.

B. Sarwar, G. Karypis, J. Konstan, and R. J. Analysis of recommendation algorithms for e-commerce. In *Proceedings ACM Electronic Commerce Conference*, pages 158--167, 2000a.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. In *ACM WebKDD Workshop*, 2000b.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW Conf.*, pages 285--295, 2001.

P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos. Collaborative filtering : Fallacies and insights in measuring similarity. In *Proceedings 10th PKDD Workshop on Web Mining (WEBMine'2006)*, pages 56--67, Berlin, 2006. URL `http://delab.csd.auth.gr/papers/WEBMINING06.pdf`.

G. Xue, C. Lin, Q. Yang, and al. et. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings ACM SIGIR Conference*, pages 114--121, 2005.