

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



Πτυχιακή Εργασία

«Αξιοποιώντας Συνεχούς Ροής Δεδομένα για την
Δημιουργία ενός Σχεδόν Πραγματικού Χρόνου
Συστήματος Συστάσεων»

Πασχάλης Γρίβας - Π2017082

Επιβλέπουσα: Ελένη Χριστοπούλου

Κέρκυρα, Σεπτέμβριος 2021



«Αξιοποιώντας Συνεχούς Ροής Δεδομένα για την
Δημιουργία ενός Σχεδόν Πραγματικού Χρόνου
Συστήματος Συστάσεων»

Πασχάλης Γρίβας - Π2017082

Τριμελής Επιτροπή

Ελένη Χριστοπούλου,
ΕΔΙΠ

Κωνσταντίνος Οικονόμου,
Καθηγητής

Μιχαήλ Στεφανιδάκης,
Αναπληρωτής Καθηγητής

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Περίληψη

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός εργαλείου παραγωγής συστάσεων, το οποίο θα ενημερώνεται αξιοποιώντας συνεχούς ροής δεδομένα και θα μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος εφαρμογών, ανεξάρτητα από το αντικείμενό τους.

Η ανάπτυξη ενός τέτοιου είδους συστήματος αποσκοπεί στο να δώσει μία λύση στο πρόβλημα, το οποίο αντιμετωπίζουν τα συστήματα συστάσεων, σχετικά με τη μείωση της ποιότητας των προτάσεών τους, καθώς αλλάζει το περιεχόμενο των εφαρμογών.

Στην πρώτη φάση της εκπόνησης της εργασίας, έγινε μία μελέτη στο θεωρητικό υπόβαθρο των συστημάτων συστάσεων προκειμένου να βρεθούν οι κατάλληλες συνιστώσες για τον ορθό σχεδιασμό ενός τέτοιου συστήματος, το οποίο επιτυγχάνει το στόχο του. Παράλληλα, πραγματοποιήθηκε έρευνα για αντίστοιχα συστήματα έτσι ώστε το νέο σύστημα να βασίζεται σε παλαιότερες μελέτες αλλά και να συνεισφέρει ουσιαστικά σε ενδεχόμενες μελλοντικές.

Στο στάδιο της ανάπτυξης, το οποίο ακολούθησε, δημιουργήθηκαν βοηθητικά προγράμματα, τα οποία προσομοιώνουν τους χρήστες και τις ενέργειες των χρηστών μιας εφαρμογής που αξιοποιεί το υπό ανάπτυξη εργαλείο. Έπειτα, αναπτύχθηκε πρόγραμμα, χρησιμοποιώντας τον κατάλληλο αλγόριθμο, τα απαραίτητα εργαλεία και τις βιβλιοθήκες, που επιλέχθηκαν κατά τον σχεδιασμό, ούτως ώστε να γίνεται η συλλογή των δεδομένων, η παραγωγή και η διανομή των συστάσεων σε μία οποιαδήποτε εφαρμογή.

Τέλος, η εργασία έχει ως αποτέλεσμα την παροχή μίας υλοποιημένης λύσης για εφαρμογές, οι οποίες αναζητούν να παρέχουν, κάθε στιγμή, ενημερωμένες και ακριβείς συστάσεις, σύμφωνα με τις τελευταίες κινήσεις των χρηστών και τις αλλαγές στο περιεχόμενό τους.

Λέξεις Κλειδιά

Συστήματα Συστάσεων, Συνεχούς Ροής Δεδομένα, Συνεργατικό Φιλτράρισμα, Alternating Least Squares, Python, Apache Kafka.

Abstract

The main goal of this thesis is the design and implementation of a tool, that generates recommendations, which will be updated, in near real time, by utilizing streaming data and can be used in a wide variety of applications, regardless of their subject matter.

The development of such a system aims to provide a simple solution to a common problem of recommendation systems which is the reduction on the quality of their recommendations as applications' content change.

At first, the theoretical background of recommendation systems was studied, in order to find the appropriate components so the system to be designed properly and achieve its goal. At the same time, a research was conducted for relevant systems, so as the new system to be based upon previous studies and also to contribute to some future ones.

In the development phase, which followed, some utility programs were developed, which role is to simulate users and their actions, in applications that utilizes this recommendation system. Then, a program was developed, using the appropriate algorithm and all the necessary tools and libraries, which were selected during the design phase, so that the collection of data and the production and distribution of recommendations to be able to take place in any application.

Finally, the project results in the provision of an implemented solution for applications, which seek to provide, up-to-date and accurate recommendations, at any time, according to the most recent user actions and changes of their content.

Keywords

Recommendation Systems, Streaming Data, Collaborative Filtering, Alternating Least Squares, Python, Apache Kafka.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτριά μου κα. Ελένη Χριστοπούλου για την εξαιρετική συνεργασία μας και τις οδηγίες της καθ' όλη τη διάρκεια εκπόνησης της πτυχιακής μου εργασίας. Επίσης, ευχαριστώ ιδιαιτέρως την οικογένειά και τους οικείους μου για την υποστήριξη και την κατανόησή τους όλο αυτό το διάστημα.

Περιεχόμενα

Περίληψη	i
Abstract	ii
Ευχαριστίες	iii
Περιεχόμενα	iv
Κατάλογος Εικόνων	vii
Κατάλογος Αποσπασμάτων Κώδικα	viii
1. Εισαγωγή	1
1.1 Κίνητρο	1
1.2 Σκοπός	2
1.3 Συνεισφορά Εργασίας	2
1.4 Διάρθρωση Εργασίας	2
2. Θεωρητικό Υπόβαθρο	3
2.1 Συστήματα Συστάσεων	3
2.2 Δεδομένα Εισόδου Συστημάτων Συστάσεων	4
2.2.1 Δεδομένα Ρητής Ανατροφοδότησης	4
2.2.2 Δεδομένα Έμμεσης Ανατροφοδότησης	4
2.3 Κατηγορίες Συστημάτων Συστάσεων	5
2.3.1 Συστήματα Φιλτραρίσματος Βάση Περιεχομένου	6
2.3.2 Συστήματα Συνεργατικού Φιλτραρίσματος	7
2.3.3 Συστήματα Υβριδικού Φιλτραρίσματος	11
2.3.4 Άλλες Κατηγορίες Συστημάτων Συστάσεων	12
2.4 Αξιολόγηση Συστημάτων Συστάσεων	12

2.4.1 Στατιστικές Μετρικές Ακρίβειας	13
2.4.2 Μετρικές Ακρίβειας Υποστήριξης Απόφασης	14
2.4.3 Άλλες Μετρικές Αξιολόγησης	15
2.5 Προκλήσεις Συστημάτων Συστάσεων	16
2.5.1 Πρόβλημα Κρύας Εκκίνησης	16
2.5.2 Υπέρ-Εξειδίκευση	16
2.5.3 Κλιμακωσιμότητα	17
2.5.4 Αραιότητα Δεδομένων	17
2.5.5 Ιδιωτικότητα	17
2.5.6 Επιθέσεις Κακόβουλης Τροφοδότησης (Shilling Attacks)	18
2.5.7 Αδυναμία Εξέλιξης Σύμφωνα με το Περιεχόμενο	19
2.6 Συνεχούς Ροής Δεδομένα	19
2.6.1 Επεξεργασία Ροών Δεδομένων	19
2.6.2 Βασικές Λειτουργίες των Stream Processing Engines	20
2.6.3 Επεξεργασία Μικρο-Ομάδων	21
2.7 Ανταλλαγή Μηνυμάτων Publish / Subscribe	21
2.8 Σύστημα Συστάσεων Πραγματικού Χρόνου	22
3. Μεθοδολογία	24
3.1 Μοντέλο Ανάπτυξης	24
3.2 Σχεδιασμός του Συστήματος	25
3.2.1 Αλγόριθμος Alternating Least Squares	27
3.3 Εργαλεία που Χρησιμοποιήθηκαν	28
3.3.1 Python 3.8	28
3.3.2 Βιβλιοθήκες της Python	29
3.3.3 Apache Kafka	30
3.3.4 Περιβάλλον Ανάπτυξης PyCharm	31
3.3.5 VMware Player	32
3.4 Απορριφθείσες Σχεδιαστικές Επιλογές και Εργαλεία	32
4. Υλοποίηση	34
4.1 Αρχιτεκτονική του Συστήματος	34
4.2 Ανάλυση Κώδικα	36

4.2.1 Πρόγραμμα Παραγωγής Αντικειμένων	36
4.2.2 Πρόγραμμα Παραγωγής Αλληλεπιδράσεων Χρήστη – Αντικειμένου	36
4.2.3 Πρόγραμμα Εκπαίδευσης Μοντέλου	37
4.2.4 Πρόγραμμα Παροχής Συστάσεων	40
4.3 Ενσωμάτωση του Συστήματος	41
4.4 Δοκιμή του Συστήματος	42
5. Αξιολόγηση	46
5.1 Από τη Μεριά του Προγραμματιστή	46
5.2 Από τη Μεριά του Χρήστη	47
6. Επίλογος	48
6.1 Συμπεράσματα	48
6.2 Μελλοντικές Προεκτάσεις	49
Βιβλιογραφικές Αναφορές	50
Συντμήσεις	53
Γλωσσάρι Ξενικών Όρων	54

Κατάλογος Εικόνων

2.1	Βασικές Τεχνικές Συστάσεων	6
2.2	Πίνακας «Χρηστών – Αντικειμένων» με Βαθμολογίες 1-5	8
2.3	Απόσταση Χρηστών με Βάση τις Βαθμολογίες σε 2 Αντικείμενα	9
2.4	Μείωση Διαστάσεων με Παραγοντοποίηση Πίνακα	10
2.5	Ακρίβεια και Ανάκληση στον Πίνακα Σύγχυσης	14
2.6	Παραδείγματα Καμπυλών ROC	15
2.7	Σταθερά Παράθυρα Μήκους 5 Δευτερολέπτων	20
2.8	Κυλιόμενα Παράθυρα Μήκους 10 και Μεσοδιάστημα 5 Δευτερολέπτων	20
2.9	Παράδειγμα Μοτίβου Ανταλλαγής Μηνυμάτων Publish/Subscribe	22
3.1	Τα Στάδια Ανάπτυξης του Μοντέλου Καταρράκτη	24
3.2	Το Λογότυπο της Python	28
3.3	Το Λογότυπο του Apache Kafka	30
3.4	Το Περιβάλλον του Λογισμικού PyCharm	31
4.1	Διάγραμμα Αρχιτεκτονικής του Συστήματος Συστάσεων	34
4.2	Προσθήκη Βοηθητικών Προγραμμάτων	35
4.3	Παράμετροι του Προγράμματος Εκπαίδευσης Μοντέλου	38
4.4	Επιστροφή 5 Προτεινόμενων Αντικειμένων για τον Χρήστη “154”	41
4.5	Εκκίνηση του Εξυπηρετητή Kafka	42
4.6	Δοκιμή του Συστήματος – Παραγωγή 25 Αντικειμένων	43
4.7	Δοκιμή του Συστήματος – Παραγωγή 5 Χρηστών	44
4.8	Σφάλμα Χρήσης Βοηθητικού Προγράμματος	44
4.9	Δοκιμή του Συστήματος – Παραγωγή Αλληλεπιδράσεων	45
5.1	Λήψη Ανατροφοδότησης για την Ποιότητα των Συστάσεων – YouTube	47

Κατάλογος Αποσπασμάτων Κώδικα

4.1	Μενού του Προγράμματος Παραγωγής Αντικειμένων	36
4.2	Παραγωγή Τυχαίων Αναγνωριστικών Αντικειμένων	36
4.3	Μενού του Προγράμματος Παραγωγής Αλληλεπιδράσεων	37
4.4	Ανάθεση Τυχαίων Μοναδικών Αναγνωριστικών στους Χρήστες	37
4.5	Παραγωγή Τυχαίων Αλληλεπιδράσεων	37
4.6	Κυλιόμενο Παράθυρο Αλληλεπιδράσεων με τη Μορφή Ουράς	38
4.7	Προετοιμασία των Δεδομένων για την Εκπαίδευση του Μοντέλου	39
4.8	Κατασκευή Μοντέλου σε Διαφορετικό Thread	39
4.9	Μετατροπή των Δεδομένων σε Πίνακες Χρήστη - Αντικειμένου	39
4.10	Εκπαίδευση του Μοντέλου με τον Αλγόριθμο ALS	40
4.11	Λήψη Αιτημάτων GET για Παραγωγή Συστάσεων	40
4.12	Παραγωγή Συστάσεων με Βάση το Τελευταίο Μοντέλο	41

Κεφάλαιο 1

Εισαγωγή

ΣΤΙΣ μέρες μας, η πληροφορία, η οποία παράγεται στον παγκόσμιο ιστό από τους χρήστες του, αυξάνεται διαρκώς. Οι ιστοσελίδες στο διαδίκτυο και οι πάροχοι ψηφιακών υπηρεσιών διευρύνουν αέναα το περιεχόμενό τους. Το γεγονός αυτό, δημιουργεί ένα πολύ σημαντικό πρόβλημα για τους χρήστες του διαδικτύου, οι οποίοι πολλές φορές αποπροσανατολίζονται, λόγω των υπεράριθμων διαθέσιμων επιλογών. Μία λύση στο πρόβλημα αυτό, η οποία εφαρμόζεται ευρύτατα, είναι η ιεράρχηση και η εξατομίκευση της πληροφορίας. Στην πράξη, αυτοί οι μετασχηματισμοί της πληροφορίας γίνονται από συστημάτων συστάσεων.

Τα συστήματα συστάσεων χρησιμοποιούν διάφορες τεχνικές και αλγορίθμους προκειμένου να προτείνουν στους χρήστες περιεχόμενο, το οποίο είναι πιθανό να ανήκει στις προτιμήσεις τους. Ωστόσο, στην προσπάθειά να επιτευχθεί ο στόχος τους προκύπτει ότι τα περισσότερα συστήματα δεν έχουν σχεδιαστεί για να παράγουν συστάσεις σε πραγματικό χρόνο [1]. Το γεγονός αυτό, επηρεάζει αρνητικά την ποιότητα των συστάσεων καθώς μεταβάλλεται το περιεχόμενο των εφαρμογών, προστίθενται νέοι χρήστες ή αλλάζουν οι προτιμήσεις τους.

Επομένως, δημιουργείται η ανάγκη ύπαρξης ενός συστήματος συστάσεων, το οποίο ενημερώνεται διαρκώς ώστε να παράγει ακριβείς συστάσεις και έχει τη δυνατότητα να προσαρμόζεται σε οποιαδήποτε εφαρμογή.

1.1 Κίνητρο

Το πρόβλημα των συστημάτων συστάσεων που αναφέρθηκε παραπάνω εντοπίστηκε κατά το σχεδιασμό μίας εφαρμογής, η οποία προτείνει στους χρήστες διάφορες δραστηριότητες και δρώμενα κοντά τους, στα πλαίσια εκπόνησης της παρούσας πτυχιακής εργασίας, το 2020.

Πιο συγκεκριμένα, με βάση την τρέχουσα τοποθεσία και τις προτιμήσεις του χρήστη, η εφαρμογή θα πρότεινε κάποια καλλιτεχνικά, πολιτιστικά, κοινωνικά γεγονότα και θεάματα που διαδραματίζονται ή πρόκειται να συμβούν γύρω του. Επίσης, θα υπάρχει η δυνατότητα ο χρήστης να ορίζει το χρονικό πλαίσιο, το οποίο επιθυμεί να διαθέσει, ενώ θα μπορεί να φιλτράρει τις επιλογές που του έχουν προταθεί. Η αρχική βλέψη της εφαρμογής είναι να λειτουργεί κυρίως σε μεγάλα αστικά περιβάλλοντα, καθώς εκεί υπάρχουν περισσότερες διαθέσιμες επιλογές, λόγω του μεγαλύτερου αριθμού κατοίκων.

Ωστόσο, προκειμένου η εφαρμογή να προτείνει δραστηριότητες στο χρήστη, θα πρέπει να χρησιμοποιηθεί ένα σύστημα συστάσεων, το οποίο μπορεί να ενημερώνεται δυναμικά καθώς μεταβάλλεται το περιεχόμενο. Για παράδειγμα, όταν προστίθενται νέα επερχόμενα δρώμενα ή όταν αφαιρούνται άλλα που έχουν περατωθεί, θα πρέπει να προτείνονται ή να σταματήσουν να προτείνονται στους χρήστες αντίστοιχα. Έτσι, με αφορμή αυτή την ανάγκη, η οποία προέκυψε, δημιουργήθηκε η ιδέα για το σχεδιασμό και την υλοποίηση ενός τέτοιου συστήματος συστάσεων.

1.2 Σκοπός

Ο σκοπός της εργασίας είναι γίνει μελέτη για τη δημιουργία ενός συστήματος συστάσεων, το οποίο θα αξιοποιεί *συνεχούς ροής δεδομένα* (steaming data), προκειμένου να παράγει συστάσεις με ακρίβεια κάθε δεδομένη χρονική στιγμή. Το σύστημα θα χρησιμοποιεί τους κατάλληλους αλγορίθμους και θα σχεδιαστεί με τέτοιο τρόπο έτσι ώστε να μπορεί να χρησιμοποιηθεί σε εφαρμογές ανεξάρτητα από το περιεχόμενό τους.

1.3 Συνεισφορά Εργασίας

Με τη δημιουργία του συστήματος συστάσεων δίνεται μία ολοκληρωμένη πρόταση σε όσους έχουν αναπτύξει ή σκοπεύουν να αναπτύξουν μία εφαρμογή και επιθυμούν να παρέχουν ακριβείς και ενημερωμένες συστάσεις στους χρήστες της. Πρόκειται για ένα εργαλείο, το οποίο με σχετικά απλό τρόπο μπορεί να ενσωματωθεί σε οποιαδήποτε αρχιτεκτονική.

Στην εργασία μελετώνται αντικείμενα από διάφορους τομείς, όπως η μηχανική μάθηση και η επεξεργασία συνεχών ροών δεδομένων, προκειμένου να σχεδιαστεί και να υλοποιηθεί το προαναφερθέν εργαλείο.

1.4 Διάρθρωση Εργασίας

Η δομή της πτυχιακής εργασίας αποτελείται από κεφάλαια, τα οποία αποτυπώνουν ως ένα βαθμό την διαδικασία που ακολουθήθηκε για τη δημιουργία του τελικού εργαλείου, ξεκινώντας από την μελέτη του θεωρητικού υποβάθρου, το σχεδιασμό, την υλοποίησή και τέλος την αξιολόγησή του.

Έτσι, στο κεφάλαιο 2, το οποίο ακολουθεί, αρχικά γίνεται μελέτη της θεωρίας των συστημάτων συστάσεων και πιο συγκεκριμένα καλύπτονται οι κατηγορίες, στις οποίες κατατάσσονται με βάση την τεχνική που χρησιμοποιούν, τα είδη των δεδομένων εισόδου, οι τρόποι και οι μετρικές που χρησιμοποιούνται προκειμένου να αξιολογούνται οι συστάσεις τους και ορισμένες συχνές προκλήσεις που συναντούν και καλούνται να αντιμετωπίσουν. Στη συνέχεια, μελετάται η επεξεργασία των συνεχών ροών δεδομένων και το μοντέλο ανταλλαγής μηνυμάτων “Publish – Subscribe”, έτσι ώστε να επιτευχθεί ο στόχος της εργασίας, η σχεδίαση και υλοποίηση του δυναμικού συστήματος συστάσεων. Τέλος, γίνεται αναφορά σε ήδη υπάρχοντα συστήματα, τα οποία εξυπηρετούν παρόμοιους σκοπούς, και στους τρόπους με τους οποίους το επιτυγχάνουν.

Το κεφάλαιο 3 αφορά τη μεθοδολογία που ακολουθήθηκε προκειμένου να δημιουργηθεί το επιθυμητό σύστημα. Πιο συγκεκριμένα, περιγράφεται το μοντέλο ανάπτυξης, το οποίο χρησιμοποιήθηκε, οι σχεδιαστικές επιλογές που έγιναν καθώς και όλα τα εργαλεία, γλώσσες προγραμματισμού, βιβλιοθήκες, περιβάλλοντα ανάπτυξης και άλλα λογισμικά που χρησιμοποιήθηκαν κατά την υλοποίηση, η οποία αναλύεται στο επόμενο κεφάλαιο. Επομένως, στο κεφάλαιο 4 περιγράφεται η αρχιτεκτονική του συστήματος, όπως προέκυψε από το σχεδιασμό του, και αναλύεται ο κώδικας των επιμέρους προγραμμάτων από τα οποία αποτελείται.

Έπειτα, στο κεφάλαιο 5 γίνεται αξιολόγηση του εργαλείου, το οποίο υλοποιήθηκε, από την σκοπιά ενός προγραμματιστή, αναφέροντας τα θετικά και τα αρνητικά του σημεία, όσον αφορά την απόδοσή του και τον τρόπο με τον οποίο δουλεύει. Επίσης, περιγράφονται τρόποι με τους οποίους θα μπορούσε το εργαλείο αυτό να αξιολογηθεί από τους τελικούς χρήστες των εφαρμογών.

Τέλος, το τελευταίο κεφάλαιο (κεφάλαιο 6) περιλαμβάνει μία σύνοψη όλων όσων αναφέρθηκαν, τα συμπεράσματα, τα οποία προκύπτουν από την συνολική ερευνητική διαδικασία και την διαδικασία ανάπτυξης, καθώς και υποθέσεις για μελλοντική έρευνα. Στο τελευταίο μέρος της εργασίας βρίσκονται όλες οι βιβλιογραφικές αναφορές και άλλες πηγές που χρησιμοποιήθηκαν.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

ΣΤΟ κεφάλαιο αυτό παρουσιάζεται αναλυτικά το θεωρητικό πλαίσιο πίσω από τα συστήματα συστάσεων. Πιο συγκεκριμένα αναλύονται, οι κατηγορίες στις οποίες κατατάσσονται, οι διαφοροποιήσεις τους με βάση το είδος των δεδομένων εισόδου, τα διάφορα προβλήματα, τα οποία προκύπτουν ανά κατηγορία και οι μέθοδοι αξιολόγησής τους. Επίσης, περιγράφεται το θεωρητικό πλαίσιο πίσω από τα συνεχούς ροής δεδομένα και την επεξεργασία τους καθώς και το μοντέλο ανταλλαγής μηνυμάτων *Publish – Subscribe*. Οι πληροφορίες αυτές αποτελούν τη βάση για τη δημιουργία ενός σχεδόν πραγματικού χρόνου συστήματος συστάσεων, το οποίο αξιοποιεί συνεχούς ροής δεδομένα.

2.1 Συστήματα Συστάσεων

Οι άνθρωποι βασίζονται σε συστάσεις από διάφορες πηγές για να επιτύχουν ένα σκοπό, δεδομένου ότι οι επιλογές, οι οποίες είναι διαθέσιμες σε αυτούς, αυξάνονται με πολύ μεγάλο ρυθμό από μέρα σε μέρα [2]. Για παράδειγμα, κάποιος ο οποίος θέλει να αγοράσει μία κινητή συσκευή, διατρέχει ένα μεγάλο αριθμό ιστοσελίδων προκειμένου να ελέγξει τις κριτικές των άλλων χρηστών ή λαμβάνει υπόψιν του τις προτάσεις των ανθρώπων, οι οποίοι τον περιβάλλουν. Αντίστοιχα, το ίδιο συμβαίνει και στις περιπτώσεις όπου κάποιος επιθυμεί να αγοράσει ένα αυτοκίνητο, να δει μία ταινία ή να διαβάσει ένα βιβλίο. Έτσι, δεδομένου ότι τα συστήματα συστάσεων βασίζονται σε αυτές τις παρατηρήσεις, προκύπτει ότι βοηθούν στη λήψη αποφάσεων, αναδεικνύοντας τις πιο κατάλληλες επιλογές, χωρίς να χρειάζεται ο χρήστης να πραγματοποιήσει αυτού του είδους την έρευνα.

Στη γενική τους μορφή, τα συστήματα συστάσεων είναι μία μέθοδος και ένα εργαλείο λογισμικού, το οποίο παρέχει τις καλύτερες προτάσεις για προϊόντα και υπηρεσίες σε χρήστες, μέσα από ένα ευρύ πεδίο δυναμικά παραγόμενης πληροφορίας [3]. Οι προτάσεις αυτές, βασίζονται στα ενδιαφέροντα και στις προτιμήσεις (προφίλ) του χρήστη μεταξύ άλλων διαφορετικών εναλλακτικών και παρουσιάζονται σε αυτόν με τον κατάλληλο τρόπο. Με άλλα λόγια, τα συστήματα συστάσεων ανήκουν σε μία ομάδα συστημάτων φιλτραρίσματος πληροφορίας, έχοντας ως στόχο την πρόβλεψη της προτίμησης ή της βαθμολογίας, την οποία έδωσε ένας χρήστης σε ένα αντικείμενο [4].

Με μαθηματικούς όρους, ένα σύστημα συστάσεων, μπορεί να οριστεί ως μία συλλογή S με όλα τα αντικείμενα, τα οποία μπορούν να προταθούν σε ένα σύνολο χρηστών U , για τους οποίους το σύστημα παράγει συστάσεις, και μία συνάρτηση F , η οποία χρησιμοποιείται για την μέτρηση της χρησιμότητας ενός συγκεκριμένου αντικειμένου $s \in S$, για έναν συγκεκριμένο χρήστη $u \in U$. Η συνάρτηση αυτή, αναπαρίσταται ως $F: U \times S \rightarrow R$, όπου R το σύνολο των συνολικών συστάσεων. Για κάθε χρήστη $u \in U$, το σύστημα προτείνει αντικείμενα $s' \in S$ τέτοια ώστε να μεγιστοποιείται η F για τον χρήστη [5].

Ένα βασικό πρόβλημα, το οποίο μπορούν να αντιμετωπίσουν τα συστήματα συστάσεων, είναι η εύρεση και η λήψη, από τους χρήστες, σχετικών και προσωποποιημένων πληροφοριών μέσα από τον τεράστιο και συνεχώς αυξανόμενο όγκο πληροφοριών στο διαδίκτυο. Παρέχοντας συστάσεις για αντικείμενα συναφή με

τον κάθε χρήστη, οι χρήστες μπορούν να επικεντρώνονται στο στόχο τους, καθώς έχουν εξοικονομήσει το χρόνο, τον οποίο θα αφιέρωναν ψάχνοντας για αυτά τα αντικείμενα.

Τα συστήματα συστάσεων παίζουν καθοριστικό ρόλο στη διευκόλυνση της λήψης αποφάσεων, γεγονός το οποίο οδήγησε στην ανάπτυξη πολλών εφαρμογών στον πραγματικό κόσμο. Η ευρεία εφαρμογή τους αποδεικνύεται από την ενσωμάτωσή τους σε διάφορους τομείς, όπως το ηλεκτρονικό εμπόριο, τις ηλεκτρονικές υπηρεσίες ψυχαγωγίας, την ηλεκτρονική διακυβέρνηση, τον ηλεκτρονικό τουρισμό κ.α. [3]. Επίσης, η μεγάλη αξία των συστημάτων αποτυπώνεται και από τα πολλαπλά οφέλη, τα οποία αποφέρουν, τόσο στους παρόχους υπηρεσιών όσο και στους πελάτες τους. Για παράδειγμα, ένας χρήστης, οποίος επισκέπτεται ένα ηλεκτρονικό κατάστημα για να αγοράσει ένα συγκεκριμένο προϊόν, είναι πιθανό να μην μπει καθόλου στη διαδικασία αναζήτησης κάποιου άλλου, πιθανόν καλύτερου, προϊόντος, με αποτέλεσμα να μην απολάβει την μέγιστη δυνατή αξία και ο ιδιοκτήτης τα επιπλέον έσοδα, τα οποία θα προέκυπταν.

2.2 Δεδομένα Εισόδου Συστημάτων Συστάσεων

2.2.1 Δεδομένα Ρητής Ανατροφοδότησης

Τα συστήματα συστάσεων βασίζονται σε διαφορετικού τύπου δεδομένα εισόδου. Το πιο πρακτικό είδος δεδομένων, το οποίο χρησιμοποιείται, είναι η *ρητή ανατροφοδότηση* (explicit feedback) των χρηστών για τα διάφορα αντικείμενα, σχετικά με την προτίμησή τους. Το είδος αυτό, μπορεί να έχει τη μορφή βαθμολογίας, με κλίμακα από 1-5 ή διαδική μορφή, θετικής ή αρνητικής πρόθεσης (π.χ. μέσω κατάλληλων κουμπιών). Ωστόσο, το βασικό πρόβλημα με τα δεδομένα τύπου ρητής ανατροφοδότησης είναι ότι δεν είναι πάντα διαθέσιμα. Προκειμένου να γίνει η συλλογή των δεδομένων, οι πάροχοι υπηρεσιών πρέπει να σχεδιάσουν σελίδες ή τμήματα ανατροφοδότησης ώστε να λαμβάνουν τις βαθμολογίες των πελατών, πράγμα το οποίο είναι χρονοβόρο και δύσκολο, διότι εξαρτάται από τον βαθμό κινητοποίησης των χρηστών. Επομένως, σε αυτές τις περιπτώσεις, προκύπτει η ανάγκη για χρήση ενός άλλου είδους «έμμεσων» δεδομένων, με υψηλότερη διαθεσιμότητα κάθε χρονική στιγμή, τα οποία να αντικατοπτρίζουν τις προτιμήσεις των χρηστών [6].

2.2.2 Δεδομένα Έμμεσης Ανατροφοδότησης

Λύση στο πρόβλημα, των δεδομένων τύπου ρητής ανατροφοδότησης, βρίσκουν τα συστήματα συστάσεων, με την χρήση δεδομένων εισόδου *έμμεσης ανατροφοδότησης* (implicit feedback), τα οποία έμμεσα αντικατοπτρίζουν τις προτιμήσεις των χρηστών [7]. Σε αυτού του είδους τα δεδομένα περιλαμβάνονται το ιστορικό αγορών, το ιστορικό περιήγησης σε ιστοσελίδες του διαδικτύου, αλληλεπιδράσεις με πολυμέσα και στοιχεία εντός των ιστοσελίδων ή ακόμα και οι κινήσεις του κέρσora των χρηστών. Για παράδειγμα, ένας χρήστης, ο οποίος αλληλεπιδρά σε μεγάλη συχνότητα με ταινίες ενός είδους, είναι πολύ πιθανό ότι του αρέσει το συγκεκριμένο είδος ταινιών.

Η πλειοψηφία της βιβλιογραφίας, στον τομέα των συστημάτων συστάσεων, επικεντρώνεται στη χρήση δεδομένων εισόδου ρητής ανατροφοδότησης, πιθανώς για πρακτικούς λόγους ευκολίας στη χρήση αυτού του είδους των πληροφοριών. Ωστόσο, σε πολλές περιπτώσεις υπάρχει η ανάγκη, τα συστήματα συστάσεων, να χρησιμοποιούν ως βάση δεδομένα έμμεσης ανατροφοδότησης. Το γεγονός αυτό, ανταποκρίνεται σε περιπτώσεις, όπου υπάρχει απροθυμία των χρηστών στο να παρέχουν αξιολογήσεις, ή σε περιπτώσεις στις οποίες υπάρχουν περιορισμοί του συστήματος, οι οποίοι αποτρέπουν τη συλλογή δεδομένων ρητής ανατροφοδότησης. Σε ένα σύστημα με δεδομένα έμμεσης ανατροφοδότησης, δεν απαιτείται καμία επιπρόσθετη ενέργεια από τους χρήστες (π.χ. παροχή αξιολογήσεων), δεδομένου ότι έχει προηγηθεί η έγκρισή τους.

Από την άλλη μεριά, η φύση των δεδομένων έμμεσης ανατροφοδότησης, προκαλεί προβλήματα στην ανάλυσή και την ερμηνεία τους, αποτρέποντας τη χρήση πολλών αλγορίθμων, οι οποίοι έχουν σχεδιαστεί

βάση των δεδομένων ρητής ανατροφοδότησης. Ένα χαρακτηριστικό των δεδομένων έμμεσης ανατροφοδότησης, το οποίο δημιουργεί προβλήματα, είναι το γεγονός ότι δεν εμπεριέχουν αρνητική ανατροφοδότηση. Παρακολουθώντας τις αλληλεπιδράσεις των χρηστών με διάφορα αντικείμενα, συμπεραίνεται ότι τα αντικείμενα αυτά πιθανώς να υπάρχουν στις προτιμήσεις τους και άρα να επιλέγουν να τα καταναλώσουν. Παρόλα αυτά, είναι δύσκολο να διαπιστωθεί με βεβαιότητα, ποια αντικείμενα δεν αρέσουν σε ένα χρήστη. Παραδείγματος χάρη, ένας χρήστης, ο οποίος δεν έχει παρακολουθήσει – αλληλεπιδράσει με μία ταινία, είναι πιθανό να μην του αρέσει η συγκεκριμένη ταινία ή απλά να μην τη γνώριζε ή να μην ήταν διαθέσιμη σε αυτόν έως τη χρονική στιγμή της συλλογής. Αυτό προκαλεί διάφορες επιπλοκές και είναι κάτι, το οποίο δεν συναντάται στα δεδομένα ρητής ανατροφοδότησης, καθώς σε αυτή την περίπτωση ο χρήστης δηλώνει ρητά τι του αρέσει και τι όχι. Τα συστήματα συστάσεων, τα οποία χρησιμοποιούν δεδομένα τύπου ρητής ανατροφοδότησης, συνήθως επικεντρώνονται στα ζεύγη χρήστη – αντικειμένου, για τα οποία έχει δοθεί και είναι γνωστή η τιμή της βαθμολογίας, αποκτώντας έτσι μία ισορροπημένη εικόνα για τις προτιμήσεις του χρήστη. Τα υπόλοιπα ζεύγη χρήστη – αντικειμένου, τα οποία συνήθως αποτελούν την πλειοψηφία των δεδομένων, αντιμετωπίζονται ως *ελλείποντα στοιχεία* (missing data) και παραλείπονται από την ανάλυση. Ωστόσο, αυτό είναι αδύνατο με τα δεδομένα έμμεσης ανατροφοδότησης, καθώς εστιάζοντας μόνο στα δεδομένα, τα οποία έχουν συλλεχθεί, δηλαδή μόνο στη θετική ανατροφοδότηση, διαστρεβλώνεται σημαντικά το πλήρες προφίλ του χρήστη. Επομένως, είναι αναγκαίο να συμπεριλαμβάνονται τα δεδομένα τα οποία λείπουν, καθώς προσεγγίζουν περισσότερο από οτιδήποτε άλλο την αρνητική ανατροφοδότηση παρότι εμπεριέχουν μεγάλη πιθανότητα σφάλματος.

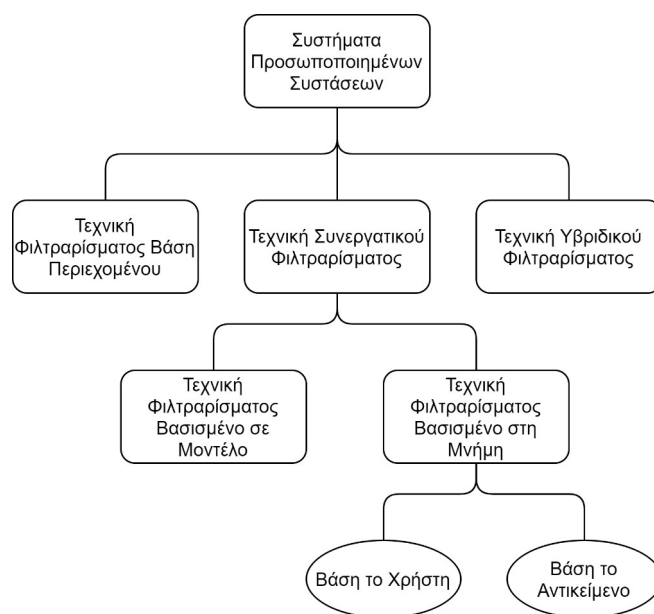
Ένα άλλο αρνητικό εγγενές χαρακτηριστικό, το οποίο έχουν τα δεδομένα έμμεσης ανατροφοδότησης, είναι ο *θόρυβος* (noise), τον οποίο περιλαμβάνουν. Με την παρακολούθηση της συμπεριφοράς των χρηστών, το μόνο που μπορεί να επιτευχθεί, είναι η εκτίμηση των προτιμήσεων και των πραγματικών κινήτρων τους. Για παράδειγμα, μπορεί να παρακολουθείται το τι αγοράζει ένας χρήστης, αλλά αυτό να μην αντιστοιχίζεται απαραίτητα σε μία θετική εικόνα του χρήστη για τα προϊόντα αυτά, καθώς μπορεί να έμεινε δυσαρεστημένος ή να μην προορίζονται για αυτόν.

Επίσης, ένας παράγοντας, ο οποίος καθιστά αδύνατη τη χρήση αλγορίθμων σχεδιασμένων για δεδομένα ρητής ανατροφοδότησης, είναι το γεγονός ότι η αριθμητική τιμή στα δεδομένα έμμεσης ανατροφοδότησης υποδηλώνει την *εμπιστοσύνη* (confidence), σε αντίθεση με την τιμή των δεδομένων ρητής ανατροφοδότησης, η οποία αντικατοπτρίζει την *προτίμηση* (preference).

Οι αριθμητικές τιμές των δεδομένων έμμεσης ανατροφοδότησης περιγράφουν την συχνότητα των ενεργειών. Χαρακτηριστικό παράδειγμα αποτελεί, το πόσο συχνά ένας χρήστης αγοράζει ένα συγκεκριμένο προϊόν, πόσες φορές ή για πόση ώρα ένας χρήστης παρακολουθούσε μία ταινία. Σε αυτές τις περιπτώσεις, μία μεγαλύτερη τιμή δε σηματοδοτεί απαραίτητα υψηλότερη προτίμηση. Για παράδειγμα, η αγαπημένη ταινία ενός χρήστη είναι κάτι το οποίο ο χρήστης έχει δει μόνο μία φορά, ενώ μπορεί να παρακολουθεί μία σειρά εβδομαδιαία, η οποία δεν του αρέσει το ίδιο. Παρόλα αυτά, η αριθμητική τιμή στα δεδομένα είναι σημαντική, καθώς εκφράζει το βαθμό «εμπιστοσύνης», ο οποίος υπάρχει σε μία αλληλεπίδραση. Ένα γεγονός, το οποίο καταγράφηκε μόνο μία φορά είναι πιθανό να μην ανταποκρίνεται στις προτιμήσεις του χρήστη, σε αντίθεση με ένα επαναλαμβανόμενο.

2.3 Κατηγορίες Συστημάτων Συστάσεων

Μία βασική κατηγοριοποίηση των συστημάτων συστάσεων αποτελεί η διάκρισή τους με βάση το είδος των συστάσεων, τα οποία παράγουν, και πιο συγκεκριμένα αν οι συστάσεις αυτές είναι *προσωποποιημένες* (Personalized) ή *μη-προσωποποιημένες* (Non-personalized). Τα συστήματα, τα οποία παράγουν μη-προσωποποιημένες συστάσεις, προτείνουν αντικείμενα στους χρήστες βασιζόμενα στον μέσο όρο των απόψεων των άλλων χρηστών. Επομένως, οι συστάσεις είναι ίδιες για όλους τους χρήστες και δεν είναι βασισμένες στις διαφορετικές προτιμήσεις του καθενός ξεχωριστά. Αντίθετα, τα συστήματα προσωποποιημένων συστάσεων χρησιμοποιούνται για να παράγουν εξατομικευμένες συστάσεις για κάθε χρήστη, βασιζόμενα στις ενέργειές τους στο παρελθόν ή/και στις αξιολογήσεις τους [4].



Εικόνα 2.1: Βασικές Τεχνικές Συστάσεων [8]

Προκειμένου τα συστήματα συστάσεων να παρέχουν προσωποποιημένες συστάσεις, είναι αναγκαίο να αναλύουν και να μετασχηματίζουν την πληροφορία, με την οποία τροφοδοτούνται, έτσι ώστε να εξάγουν συμπεράσματα για τις προτιμήσεις των χρηστών και για τα αντικείμενα εκείνα, τα οποία υπάγονται σε αυτές. Ωστόσο, το είδος της πληροφορίας και ο τρόπος, με τον οποίο αξιοποιείται, διαφέρει ανάλογα με την τεχνική συστάσεων που εφαρμόζεται [Εικόνα 2.1]. Έτσι, τα συστήματα συστάσεων ταξινομούνται σε διάφορες κατηγορίες όπως τα Συστήματα Συνεργατικού Φιλτραρίσματος και τα Συστήματα Φιλτραρίσματος Βάση Περιεχομένου.

2.3.1 Συστήματα Φιλτραρίσματος Βάση Περιεχομένου

Μία τεχνική, η οποία χρησιμοποιείται στα σύστημα συστάσεων, προκειμένου να παρέχουν ακριβείς και ενδιαφέρουσες για τους χρήστες συστάσεις, είναι το *Φιλτράρισμα Βάση Περιεχομένου* (Content-Based Filtering). Η βασική ιδέα, στην οποία στηρίζονται τα συστήματα που υλοποιούν την τεχνική φιλτραρίσματος με βάση το περιεχόμενο, είναι να προτείνουν αντικείμενα σε ένα χρήστη, τα οποία είναι παρόμοια με αυτά που του είχαν αρέσει στο παρελθόν. Ο υπολογισμός της ομοιότητας των αντικειμένων, βασίζεται στην ομοιότητα των χαρακτηριστικών τους. Με άλλα λόγια, η τεχνική στηρίζεται στην ανάλυση του περιεχομένου των αντικειμένων, δηλαδή των διάφορων χαρακτηριστικών τους, έτσι ώστε να παράγονται προβλέψεις για νέα αντικείμενα [8].

Ένα σύστημα φιλτραρίσματος με βάση το περιεχόμενο, χρησιμοποιεί διαφορετικών ειδών μοντέλα, με σκοπό να εκτιμήσει την ομοιότητα μεταξύ των αντικειμένων. Ανάμεσα σε αυτά τα μοντέλα μπορεί να βρίσκονται διάφορα *πιθανοτικά μοντέλα* (Probabilistic models), όπως ο “Naive Bayes Classifier”, τα Δέντρα Απόφασης ή τα νευρωνικά δίκτυα, προκειμένου να μοντελοποιηθεί η σχέση μεταξύ των διάφορων αντικειμένων στο σύστημα. Αυτές οι μέθοδοι, παράγουν συστάσεις βασιζόμενες σε ένα μοντέλο κατασκευασμένο από τα υποκείμενα δεδομένα, χρησιμοποιώντας τεχνικές στατιστικής ανάλυσης ή τεχνικές μηχανικής μάθησης. Με άλλα λόγια, τα αντικείμενα κατηγοριοποιούνται αυτοματοποιημένα σε σχετικά και μη-σχετικά, γίνεται δηλαδή η πρόβλεψη της κλάσης, συγκρίνοντας τα χαρακτηριστικά των αντικειμένων του συστήματος με τα χαρακτηριστικά των αντικειμένων που ανήκουν στις προτιμήσεις του χρήστη.

Ένα πλεονέκτημα των συστημάτων φιλτραρίσματος με βάση το περιεχόμενο, είναι το γεγονός ότι δεν απαιτούν τα προφίλ των άλλων χρηστών, καθώς αυτά δεν εμπλέκονται στη διαδικασία παραγωγής συστάσεων. Επίσης, ένα πλεονέκτημα της συγκεκριμένης τεχνικής αποτελεί ο τρόπος ανταπόκρισης των συστημάτων στην περίπτωση μιας αλλαγής στο προφίλ του χρήστη, καθώς έχουν τη δυνατότητα να

προσαρμόζουν τις συστάσεις σε πολύ σύντομο χρονικό διάστημα. Επιπλέον, η τεχνική φιλτραρίσματος με βάση το περιεχόμενο, μπορεί να συστήνει καινούρια αντικείμενα, ακόμα και αν δεν υπάρχει κανενός είδους ανατροφοδότησης από τους χρήστες, να ανταπεξέρχεται σε περιπτώσεις όπου οι χρήστες δεν έχουν μεταξύ τους κοινά αντικείμενα στις προτιμήσεις τους, ενώ μπορούν να παρέχουν επεξηγήσεις για τις συστάσεις, τις οποίες κάνουν (π.χ. «επειδή παρακολουθήσατε μία συγκεκριμένη ταινία»). Τέλος, με αυτή την τεχνική, οι χρήστες λαμβάνουν συστάσεις χωρίς να μοιράζονται το προφίλ προτιμήσεών τους, γεγονός το οποίο ενισχύει την ιδιωτικότητά τους.

Από την άλλη μεριά, υπάρχουν διάφορα προβλήματα, τα οποία προκύπτουν στα συστήματα φιλτραρίσματος με βάση το περιεχόμενο, λόγω αυτής της τεχνικής. Το βασικότερο μειονέκτημά της, είναι η ανάγκη να υπάρχει μία διεξοδική γνώση και λεπτομερείς περιγραφή των χαρακτηριστικών των αντικειμένων, όπως επίσης να υπάρχουν καλά οργανωμένα προφίλ χρηστών. Επομένως, η αποτελεσματικότητα των συστημάτων και η ακρίβεια των συστάσεων, βασίζεται στη διαθεσιμότητα και στην ποιότητα των μεταδεδομένων των αντικειμένων. Ένα άλλο βασικό πρόβλημα το οποίο προκύπτει, είναι η υπερ-εξειδίκευση. Οι συστάσεις, οι οποίες γίνονται στους χρήστες, βασίζονται αποκλειστικά στα αντικείμενα, τα οποία ανήκουν στις προτιμήσεις τους. Επομένως, στην περίπτωση όπου ένας χρήστης αλληλεπιδρά ή έχει δηλώσει την προτίμησή του για αντικείμενα ενός είδους, δεν θα λάβει ποτέ συστάσεις για αντικείμενα άλλων ειδών, τα οποία μπορεί επίσης να του αρέσουν.

2.3.2 Συστήματα Συνεργατικού Φιλτραρίσματος

Μία άλλη βασική κατηγορία συστημάτων συστάσεων είναι τα συστήματα τα οποία χρησιμοποιούν την τεχνική του *Συνεργατικού Φιλτραρίσματος* (Collaborative Filtering). Σε αντίθεση με την τεχνική φιλτραρίσματος βάση περιεχομένου, η οποία βασίζεται στην υπόθεση ότι αντικείμενα με παρόμοια χαρακτηριστικά τείνουν να λαμβάνουν αντίστοιχες βαθμολογίες, το συνεργατικό φιλτράρισμα στηρίζεται στο ότι οι άνθρωποι με παρόμοιες προτιμήσεις αξιολογούν τα αντικείμενα με παρόμοιο τρόπο. Το φιλτράρισμα με βάση το περιεχόμενο αντιστοιχίζει τον χρήστη με τα χαρακτηριστικά των αντικειμένων, τα οποία βαθμολόγησε, και έπειτα τα συγκρίνει με τα χαρακτηριστικά νέων αντικειμένων. Ωστόσο, αυτό προϋποθέτει ότι έχει προηγηθεί ανάλυση του περιεχομένου των αντικειμένων και έχει γίνει εξαγωγή των χαρακτηριστικών τους, κάτι το οποίο πολλές φορές είναι δύσκολο ή μη εφικτό (π.χ. λόγω λιγοστού περιεχομένου ή έλλειψης μεταδεδομένων). Με άλλα λόγια, εάν δεν υπάρχει κάποιος εύκολος τρόπος ώστε να γίνει εξαγωγή ενός χαρακτηριστικού αυτοματοποιημένα, τότε αυτό δεν μπορεί να χρησιμοποιηθεί. Η τεχνική του συνεργατικού φιλτραρίσματος δεν συναντά τέτοιου είδους προβλήματα καθώς δεν απαιτείται το περιεχόμενο των αντικειμένων και η αξιολόγηση γίνεται από τους χρήστες με έμμεσο ή άμεσο τρόπο [9].

Το συνεργατικό φιλτράρισμα είναι μία τεχνική πρόβλεψης, η οποία συνήθως χρησιμοποιείται για αντικείμενα που δεν μπορούν να περιγραφούν εύκολα και σε ικανοποιητικό βαθμό από μεταδεδομένα, όπως οι ταινίες και η μουσική. Αρχικά, η τεχνική του συνεργατικού φιλτραρίσματος κατασκευάζει έναν πίνακα χρηστών - αντικειμένων, ο οποίος περιλαμβάνει τις προτιμήσεις του κάθε χρήστη για κάθε αντικείμενο [Εικόνα 2.2]. Κάθε γραμμή του πίνακα περιέχει τις βαθμολογίες ενός χρήστη και κάθε στήλη τις βαθμολογίες, τις οποίες έχει λάβει ένα αντικείμενο. Έπειτα, αντιστοιχίζει χρήστες με παρόμοια ενδιαφέροντα και προτιμήσεις υπολογίζοντας τις ομοιότητες μεταξύ των προφίλ τους προκειμένου να παραχθούν συστάσεις. Οι χρήστες οι οποίοι είναι παρόμοιοι - «βρίσκονται κοντά» και συμβάλουν ώστε να προταθούν νέα αντικείμενα σε ένα χρήστη αποτελούν την «γειτονιά» του χρήστη αυτού. Ουσιαστικά, σε έναν χρήστη συστήνονται αντικείμενα, τα οποία δεν έχει βαθμολογήσει στο παρελθόν αλλά έχουν ήδη βαθμολογηθεί θετικά από γειτονικούς του χρήστες. Οι συστάσεις που παράγονται από συστήματα, τα οποία χρησιμοποιούν αυτή την τεχνική, μπορούν να αποτελούν είτε πρόβλεψη είτε σύσταση. Η πρόβλεψη είναι μία αριθμητική τιμή R_{ui} , η οποία υποδηλώνει την βαθμολογία που θα έδινε ένας χρήστης u σε ένα αντικείμενο i , ενώ η σύσταση είναι μία λίστα από τα καλύτερα N αντικείμενα, τα οποία θα άρεσαν περισσότερο σε έναν χρήστη [8]. Η τεχνική του συνεργατικού φιλτραρίσματος χωρίζεται σε δύο επιμέρους κατηγορίες, *Με Βάση τη Μνήμη* (Memory-based) και *Με Βάση το Μοντέλο* (Model-based).

	i_1	i_2	i_3	i_4	i_5
u_1	5		4	1	
u_2		3		3	
u_3		2	4	4	1
u_4	4	4	5		
u_5	2	4		5	2

Εικόνα 2.2: Πίνακας «Χρηστών - Αντικειμένων» με Βαθμολογίες 1-5 [11]

Συνεργατικό Φιλτράρισμα με Βάση τη Μνήμη

Τα αντικείμενα, τα οποία έχουν βαθμολογηθεί από έναν χρήστη, είναι αυτά που συμβάλουν στην αναζήτηση ενός γειτονικού χρήστη με παρόμοιες προτιμήσεις. Η πιο διαδεδομένη “memory-based” προσέγγιση του συνεργατικού φιλτραρίσματος είναι οι “neighborhood-based” αλγόριθμοι [10]. Οι αλγόριθμοι αυτοί εστιάζουν στον υπολογισμό της σχέσης - απόστασης μεταξύ των χρηστών ή των αντικειμένων. Έχοντας βρει τους γείτονες ενός χρήστη, διάφοροι αλγόριθμοι μπορούν να χρησιμοποιηθούν προκειμένου να συνδυαστούν οι προτιμήσεις των γειτονικών χρηστών και να παραχθούν συστάσεις. Το συνεργατικό φιλτράρισμα με βάση τη μνήμη μπορεί να επιτευχθεί με δύο τρόπους, με βάση τον χρήστη (user-based) ή με βάση το αντικείμενο (item-based).

Η τεχνική συνεργατικού φιλτραρίσματος με βάση τον χρήστη υπολογίζει την ομοιότητα μεταξύ των χρηστών συγκρίνοντας τις βαθμολογίες που έχουν δώσει σε ένα αντικείμενο [Εικόνα 2.3]. Έπειτα υπολογίζει την βαθμολογία (πρόβλεψη), την οποία θα έδινε ο χρήστης για ένα αντικείμενο, ως τον σταθμισμένο μέσο όρο των βαθμολογιών των παρόμοιων χρηστών για αυτό το αντικείμενο, με βάρη την ομοιότητα των χρηστών [8]. Έτσι, η τελική πρόβλεψη για την βαθμολογία βασίζεται περισσότερο στους γειτονικούς χρήστες οι οποίοι έχουν μεγαλύτερο βαθμό ομοιότητας με τον τρέχοντα χρήστη. Με μαθηματικό τρόπο αυτό εκφράζεται με τον ακόλουθο τύπο, όπου η βαθμολογία Ru , την οποία θα έδινε ο τρέχον χρήστης σε ένα αντικείμενο, είναι το

γινόμενο της βαθμολογίας R_u του αντικειμένου από κάθε γειτονικό χρήστη u με το βαθμό ομοιότητας αυτού, προς το άθροισμα κάθε βαθμού ομοιότητας S_u [11] :

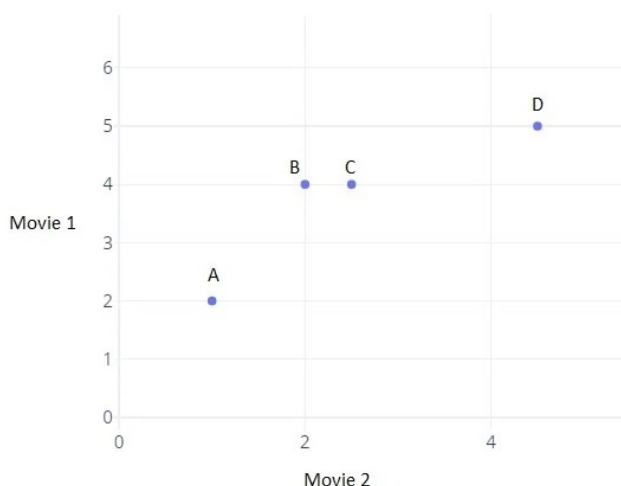
$$Ru = \left(\sum_{u=1}^n R_u * S_u \right) / \left(\sum_{u=1}^n S_u \right)$$

Από την άλλη, η τεχνική συνεργατικού φιλτραρίσματος με βάση το αντικείμενο υπολογίζει τις προβλέψεις των βαθμολογιών χρησιμοποιώντας την ομοιότητα μεταξύ των αντικειμένων και όχι την ομοιότητα μεταξύ των χρηστών. Πιο συγκεκριμένα, για ένα αντικείμενο, βρίσκει αντικείμενα τα οποία έχουν λάβει παρόμοιες βαθμολογίες από τους χρήστες και υπολογίζει την βαθμολογία που θα έδινε ο τρέχον χρήστης με βάση τις βαθμολογίες που έχει δώσει στα πιο κοντινά αντικείμενα. Η τεχνική του “item-based” φιλτραρίσματος είναι γρηγορότερη και πιο σταθερή από ότι η “user-based” σε μία εφαρμογή, στην οποία υπάρχουν περισσότεροι χρήστες από ότι αντικείμενα. Συνήθως, είναι περισσότερο αποτελεσματική διότι η μέση βαθμολογία που λαμβάνει ένα αντικείμενο δεν αλλάζει τόσο γρήγορα όσο την μέση βαθμολογία, την οποία δίνει ένας χρήστης σε διάφορα αντικείμενα [11].

Διάφοροι τύποι μετρικών ομοιότητας χρησιμοποιούνται για να υπολογίζεται η ομοιότητα μεταξύ των χρηστών ή των αντικειμένων. Ένας δημοφιλής τρόπος για τον υπολογισμό είναι η *ομοιότητα συνημιτόνου* (cosine similarity), η οποία δίνεται από τον ακόλουθο τύπο [10]:

$$\cos(\vec{x}, \vec{y}) = \frac{\sum_i r_{x,i} r_{y,i}}{\sqrt{\sum_i r_{x,i}^2} \sqrt{\sum_i r_{y,i}^2}}$$

Όπου I_{xy} το σύνολο των αντικειμένων που βαθμολογήθηκαν από τους χρήστες x και y . Άλλη μία μετρική που χρησιμοποιείται συχνά είναι ο συντελεστής συσχέτισης Pearson.



Εικόνα 2.3: Απόσταση Χρηστών με Βάση τις Βαθμολογίες σε 2 Αντικείμενα [11]

Συνεργατικό Φιλτράρισμα με Βάση το Μοντέλο

Αποσκοπώντας στην βελτίωση της τεχνικής του συνεργατικού φιλτραρίσματος, η τεχνική του συνεργατικού φιλτραρίσματος με βάση το μοντέλο χρησιμοποιεί τις βαθμολογίες των χρηστών σε αντικείμενα προκειμένου να κατασκευαστεί ένα μοντέλο. Η εκπαίδευση του μοντέλου μπορεί να επιτευχθεί χρησιμοποιώντας τεχνικές μηχανικής μάθησης, ανακαλύπτοντας μοτίβα – σχέσεις μεταξύ χρηστών και αντικειμένων στα δεδομένα εκπαίδευσης, ή τεχνικές εξόρυξης δεδομένων. Οι τεχνικές αυτές έχουν τη δυνατότητα να παράγουν συστάσεις για αντικείμενα πολύ γρήγορα λόγω του προ-υπολογισμένου μοντέλου, το οποίο χρησιμοποιούν. Έχει αποδειχθεί ότι παράγουν παρόμοια αποτελέσματα με αυτά των τεχνικών συνεργατικού φιλτραρίσματος με βάση τη μνήμη. Μερικά παραδείγματα αυτών των τεχνικών είναι η *Μείωση Διαστάσεων* (Dimensionality Reduction) όπως η *Αποσύνθεση Ιδιαζουσών Τιμών* (Singular Value Decomposition - SVD), η ανάλυση με *Γραμμική Παλινδρόμηση* (Regression) και η *Συσταδοποίηση* (Clustering) [8].

Αναλυτικότερα, μία κατηγορία αλγορίθμων που χρησιμοποιούνται στα “model-based” συστήματα συστάσεων είναι οι **Κανόνες Συσχέτισης** (Association rules). Οι αλγόριθμοι εξόρυξης κανόνων συσχέτισης εξάγουν κανόνες, οι οποίοι προβλέπουν την ύπαρξη ενός αντικειμένου με βάση την ύπαρξη άλλων αντικειμένων σε μία συναλλαγή. Για παράδειγμα, αν παρατηρείται ότι ένα αντικείμενο **A** βαθμολογείται θετικά από πολλούς χρήστες, οι οποίοι βαθμολογούν επίσης θετικά ένα αντικείμενο **B**, τότε το αντικείμενο **B** είναι πιθανό να προταθεί σε έναν χρήστη που έχει βαθμολογήσει το **A** και δεν το έχει ακόμα «καταναλώσει». Ωστόσο, παρόλο που οι κανόνες συσχέτισης έχουν κοινό στόχο με τα συστήματα συστάσεων, δεν χρησιμοποιούνται συχνά [8].

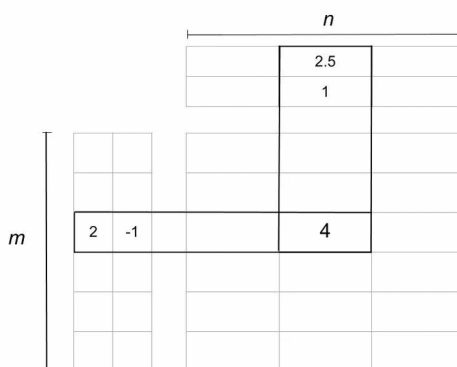
Μία άλλη τεχνική που χρησιμοποιείται είναι η **Συσταδοποίηση** ή Ομαδοποίηση. Η τεχνική αυτή έχει χρησιμοποιείται σε διάφορους τομείς, όπως την αναγνώριση μοτίβων, την επεξεργασία εικόνας, την

στατιστική ανάλυση και την εξόρυξη γνώσης. Ένας αλγόριθμος συσταδοποίησης προσπαθεί να χωρίσει ένα σύνολο δεδομένων σε ομάδες, τα μέλη των οποίων έχουν κάποια κοινά χαρακτηριστικά. Σε ένα σύστημα συστάσεων, έχοντας σχηματίσει τις ομάδες χρηστών με βάση τις προτιμήσεις τους, μπορούν να γίνουν συστάσεις σε έναν χρήστη υπολογίζοντας τον μέσο όρο των βαθμολογιών των άλλων χρηστών στην ίδια ομάδα. Οι αλγόριθμοι “K-means” και *Δίκτυα Αυτο-Οργανούμενης Απεικόνισης* (Self-Organizing Map - SOM) είναι οι πιο διαδεδομένοι όσον αφορά την τεχνική της συσταδοποίησης [8].

Αντίστοιχα με την τεχνική της συσταδοποίησης, μία άλλη τεχνική που χρησιμοποιείται για την αναγνώριση σχέσεων μεταξύ των χρηστών και των αντικειμένων είναι η ανάλυση με **Γραμμική Παλινδρόμηση**. Η τεχνική χρησιμοποιείται όταν δύο ή περισσότερες μεταβλητές θεωρείται ότι είναι συστηματικά συνδεδεμένες με μία γραμμική σχέση. Είναι μία πολύ αποτελεσματική διαδικασία για την ανάλυση σχέσεων μεταξύ μίας εξαρτημένης μεταβλητής με μία ή περισσότερες ανεξάρτητες. Η ανάλυση με γραμμική παλινδρόμηση χρησιμοποιείται για *ταίριασμα καμπύλης* (curve fitting), προβλέψεις και *έλεγχο υπόθεσης* (hypothesis systematic testing) για τις σχέσεις μεταξύ μεταβλητών. Το ταίριασμα καμπύλης είναι χρήσιμο στον εντοπισμό μίας *τάσης* (trend) εντός του συνόλου των δεδομένων, είτε πρόκειται για γραμμική, με την μορφή παραβολής ή κάποια άλλη μορφή.

Μία ακόμη τεχνική, η οποία χρησιμοποιείται στο “model-based” συνεργατικό φιλτράρισμα, είναι η **Μείωση των Διαστάσεων**. Όπως περιγράφηκε νωρίτερα σε αυτή την ενότητα, οι τεχνικές του συνεργατικού φιλτραρίσματος αναπαριστούν τα δεδομένα των βαθμολογιών ως έναν πίνακα χρήστη – αντικειμένου. Ωστόσο, τα κελιά του πίνακα αυτού τις περισσότερες φορές είναι κενά, καθώς οι χρήστες βαθμολογούν λίγα αντικείμενα σε σχέση με το σύνολο των διαθέσιμων αντικειμένων, με αποτέλεσμα ο πίνακας να είναι αραιός. Επίσης, προκύπτει, σε πολλές περιπτώσεις το μέγεθος του πίνακα να είναι μεγάλο, λόγω του μεγάλου αριθμού χρηστών ή αντικειμένων που μπορεί να υπάρχουν σε μία εφαρμογή. Το πρόβλημα αυτό οδηγεί στην αδυναμία του συστήματος να παρέχει αξιόπιστες και ακριβείς συστάσεις στους χρήστες. Έτσι, μία μείωση των διαστάσεων του πίνακα μπορεί αφενός να διορθώσει το πρόβλημα αυτό, με την ποιότητα των συστάσεων, και αφετέρου να βελτιώσει τις επιδόσεις του συστήματος όσον αφορά τον χρόνο και τον χώρο. Η μείωση των διαστάσεων μπορεί να επιτευχθεί χρησιμοποιώντας διάφορες μεθόδους όπως την *Παραγοντοποίηση Πίνακα* (Matrix Factorization) ή τους *Αυτοκωδικοποιητές* (Autoencoders), ένα είδος *Τεχνητών Νευρωνικών Δικτύων* (Artificial Neural Networks -ANN) [11].

Η παραγοντοποίηση πίνακα μπορεί να οριστεί ως η διάσπαση ενός μεγάλου πίνακα σε ένα γινόμενο μικρότερων πινάκων. Έτσι, ένας πίνακας A με διαστάσεις $m \times n$ μπορεί να «σπάσει» σε ένα γινόμενο δύο πινάκων X και Y με διαστάσεις $m \times p$ και $p \times n$ αντίστοιχα. Από τους δύο μικρότερους πίνακες [Εικόνα 2.4], οι γραμμές m του πρώτου αναπαριστούν τους χρήστες ενώ οι στήλες p αναπαριστούν τα χαρακτηριστικά αυτών των χρηστών. Αντίστοιχα οι στήλες n του δεύτερου πίνακα αντιστοιχούν στα αντικείμενα και οι γραμμές p στα χαρακτηριστικά τους [11].



Εικόνα 2.4: Μείωση Διαστάσεων με Παραγοντοποίηση Πίνακα [11]

Οι δύο στήλες του πίνακα των χρηστών και οι δύο γραμμές του πίνακα των αντικειμένων [Εικόνα 2.4], ονομάζονται *κρυφοί παράγοντες* (latent factors) και υποδεικνύουν κρυφά χαρακτηριστικά των χρηστών και

των αντικειμένων αντίστοιχα. Οι παράγοντες αυτοί είναι στην πραγματικότητα αριθμητικές τιμές και δεν μπορούν να ταυτιστούν με τα πραγματικά χαρακτηριστικά που μπορεί να έχουν οι χρήστες και τα αντικείμενα. Είναι ένα από τα πράγματα, τα οποία βελτιστοποιούνται κατά την εκπαίδευση του μοντέλου. Η διαδικασία της εκπαίδευσης, στην πραγματικότητα αποτελεί την εύρεση των κατάλληλων αριθμητικών τιμών των κρυφών παραγόντων, έτσι ώστε από το γινόμενο τους να προκύπτει, κατά προσέγγιση, ο αρχικός πίνακας, με συμπληρωμένες τις τιμές που έλλειπαν (Matrix Completion). Με αυτόν τον τρόπο, γίνεται μία εκτίμηση των βαθμολογιών των χρηστών, οι οποίες δεν έχουν ακόμη δοθεί. Τέλος, πρέπει να σημειωθεί ότι ο αριθμός των κρυφών παραγόντων – αριθμός στηλών του πίνακα των χρηστών και αριθμός γραμμών του πίνακα των αντικειμένων, επηρεάζουν την ποιότητα των παραγόμενων συστάσεων. Όσο περισσότεροι είναι οι κρυφοί παράγοντες, τόσο μεγαλύτερη θα είναι η ακρίβεια των συστάσεων. Ωστόσο, ένας υπερβολικά μεγάλος αριθμός παραγόντων μπορεί να οδηγήσει στην *υπερ-εξειδίκευση* (over-fitting) του μοντέλου, δηλαδή στο να έχει μία καλή απόδοση στα δεδομένα, με τα οποία εκπαιδεύτηκε, αλλά να έχει κακή απόδοση στα νέα δεδομένα.

Ένας πολύ δημοφιλής αλγόριθμος για παραγοντοποίηση πίνακα είναι η Αποσύνθεση Ιδιαζουσών Τιμών, ο οποίος χρησιμοποιήθηκε στον ανοιχτό διαγωνισμό του Netflix “Netflix Prize” [11]. Επίσης ένας πολύ διαδεδομένος αλγόριθμος, ο οποίος χρησιμοποιείται σε αυτή την τεχνική, είναι η *Μέθοδος Εναλλασσόμενου Ελαχίστου Τετραγώνου* (Alternating Least Square – ALS) [8]. Ο αλγόριθμος αυτός βελτιστοποιεί τους πίνακες X και Y με εναλλασσόμενο τρόπο προκειμένου να ελαχιστοποιήσει το σφάλμα από τον αρχικό πίνακα. Ανάμεσα σε άλλους αλγορίθμους, οι οποίοι επίσης χρησιμοποιούνται, βρίσκεται ο αλγόριθμος Στοχαστική Επικλινής Κάθοδος (“Stochastic Gradient Descent”), ο “Bias Stochastic Gradient Descent” και ο “Weighted Alternating Least Square” [10].

2.3.3 Συστήματα Υβριδικού Φιλτραρίσματος

Το *Υβριδικό Φιλτράρισμα* (Hybrid Filtering) είναι μία ακόμα τεχνική, την οποία χρησιμοποιούν τα συστήματα που ανήκουν σε αυτή την κατηγορία. Το υβριδικό φιλτράρισμα συνδυάζει διαφορετικές τεχνικές παραγωγής συστάσεων προκειμένου να αποφευχθούν ορισμένοι περιορισμοί και προβλήματα των συστημάτων συστάσεων. Η βασική θεωρία, στην οποία στηρίζεται αυτή η τεχνική είναι, το γεγονός ότι ο συνδυασμός αλγορίθμων παράγει καλύτερα αποτελέσματα όσον αφορά την ακρίβεια των συστάσεων, από έναν αλγόριθμο που χρησιμοποιείται μεμονωμένα [1, 2, 8, 12]. Χρησιμοποιώντας έτσι πολλαπλές τεχνικές εξαλείφονται οι αδυναμίες των επιμέρους αλγορίθμων σε ένα συνδυαστικό μοντέλο. Οι τεχνικές του υβριδικού φιλτραρίσματος μπορούν να χωριστούν σε τρεις υποκατηγορίες με βάση τον τρόπο με τον οποίο δουλεύουν [2].

Τα *Μονολιθικά Υβριδικά Συστήματα* (Monolithic Hybrid Systems) είναι μία από τις υποκατηγορίες των υβριδικών συστημάτων και αποτελούνται από ένα μοναδικό σκέλος, το οποίο συνδυάζει τις δύο ή περισσότερες τεχνικές, προ-υπολογίζοντας διάφορες πηγές δεδομένων. Τα συστήματα ταξινομούνται σε *συστήματα συνδυασμού χαρακτηριστικών* (feature combination systems) και *συστήματα επαύξησης χαρακτηριστικών* (feature augmentation systems). Τα πρώτα, χρησιμοποιούν πολλαπλές πηγές δεδομένων ως είσοδο ενός αλγορίθμου παραγωγής συστάσεων, ενώ τα δεύτερα χρησιμοποιούν την έξοδο του ενός αλγορίθμου ως είσοδο για τον επόμενο [12].

Άλλη υποκατηγορία υβριδικών συστημάτων είναι τα *Παραλληλοποιημένα Υβριδικά Συστήματα* (Parallelized Hybrid Systems). Τα συστήματα αυτά αποτελούνται από πολλαπλά συστήματα συστάσεων ταυτόχρονα, τα αποτελέσματα των οποίων συνδυάζονται με έναν συγκεκριμένο μηχανισμό υβριδισμού. Στα παραλληλοποιημένα συστήματα κατατάσσονται τα *μεικτά* (mixed), τα *εναλλασσόμενα* (switching) και τα υβριδικά συστήματα *με βάρη* (weighted). Τα υβριδικά συστήματα με βάρη, παράγουν συστάσεις συμπεριλαμβάνοντας τα σταθμισμένα αθροίσματα των αποτελεσμάτων δύο ή περισσότερων συστημάτων συστάσεων σε μία μοναδική λίστα συστάσεων. Αντίστοιχα, τα μεικτά υβριδικά συστήματα, με παρόμοιο τρόπο με τα συστήματα με βάρη, συνδυάζουν τα αποτελέσματα διάφορων συστημάτων συστάσεων. Τέλος, τα εναλλασσόμενα υβριδικά συστήματα, βασίζονται στην ποιότητα των παραγόμενων συστάσεων και στις απαιτήσεις των χρηστών, εναλλάσσουν διάφορους αλγορίθμους ώστε να παραχθούν τα καλύτερα δυνατά αποτελέσματα [2].

Στην τελευταία υποκατηγορία των υβριδικών συστημάτων ανήκουν τα *Διασυνδεδεμένα Υβριδικά Συστήματα* (Pipelined Hybrid System). Τα pipelined συστήματα παράγουν συστάσεις ακολουθώντας μία διαδικασία κατά την οποία πολλαπλές τεχνικές παραγωγής συστάσεων χρησιμοποιούν η μία την άλλη. Σε αυτά τα συστήματα κατατάσσονται τα υβριδικά συστήματα *καταρράκτη* (cascade), τα οποία τροφοδοτούν τα αποτελέσματα ενός αλγορίθμου ως είσοδο ενός άλλου, και τα υβριδικά συστήματα *μετα-επιπέδου* (meta-level), τα οποία εκπαιδεύουν ένα μοντέλο χρησιμοποιώντας έναν αλγόριθμο και έπειτα το χρησιμοποιούν ως είσοδο σε έναν άλλο [12].

2.3.4 Άλλες Κατηγορίες Συστημάτων Συστάσεων

Στις προηγούμενες υποενότητες περιγράφηκαν οι βασικές κατηγορίες των συστημάτων συστάσεων, με βάση την τεχνική την οποία χρησιμοποιούν, συνεργατικό φιλτράρισμα, φιλτράρισμα με βάση το περιεχόμενο και υβριδικό φιλτράρισμα. Ωστόσο, υπάρχουν ακόμα πολλές τεχνικές που επίσης χρησιμοποιούνται, συνθέτοντας επιπλέον κατηγορίες συστημάτων συστάσεων.

Ανάμεσα σε αυτές τις κατηγορίες βρίσκονται τα *Context-Aware Συστήματα Συστάσεων* (Context-Aware Recommender Systems). Τα περισσότερα συστήματα συστάσεων εστιάζουν στη συλλογή των δεδομένων, τα οποία έχουν την μορφή ενός πίνακα δύο διαστάσεων ή στην ανάλυση των μεταδεδομένων των αντικειμένων, προκειμένου να γίνει εξαγωγή συμπερασμάτων για τις προτιμήσεις των χρηστών. Ωστόσο, τα context-aware συστήματα πέρα από τη διαδικασία ανάλυσης των δεδομένων αυτών, εκμεταλλεύονται την *πληροφορία των συνθηκών* (context), προκειμένου να παράγουν καλύτερες, περισσότερο προσωποποιημένες συστάσεις στους χρήστες [3]. Πιο συγκεκριμένα, οι πληροφορίες για τις συνθήκες των χρηστών μπορεί να περιλαμβάνουν την τοποθεσία του χρήστη, την τρέχουσα ημερομηνία, εποχή, θερμοκρασία, διάθεση κ.ο.κ.. Οι πληροφορίες αυτές μπορούν να συλλέγονται ρητά από τον χρήστη (π.χ. ρωτώντας για την διάθεση του χρήστη ώστε να του προταθεί μία ανάλογη μουσική ή η κατάλληλη ταινία), ή να λαμβάνονται έμμεσα από το περιβάλλον του χρήστη (π.χ. η τρέχουσα εποχή για να του συσταθούν τα κατάλληλα προϊόντα ένδυσης) [4].

Τα *Συστήματα Συστάσεων Δημογραφικού Φιλτραρίσματος* (Demographic Filtering Systems) είναι μία ακόμα κατηγορία συστημάτων συστάσεων. Χρησιμοποιούν τα δημογραφικά δεδομένα των χρηστών, τα οποία έχουν εκ των προτέρων συλλεχθεί, και στις απόψεις των χρηστών για αντικείμενα, τα οποία τους έχουν προταθεί. Τα δημογραφικά συστήματα βασίζονται στην υπόθεση ότι όλοι οι χρήστες, που ανήκουν σε μία ομάδα χρηστών με κοινά δημογραφικά χαρακτηριστικά, έχουν παρόμοιες προτιμήσεις [2].

Το μεγάλο πλεονέκτημα αυτών των συστημάτων είναι το γεγονός ότι δεν χρειάζονται δεδομένα βαθμολογιών των χρηστών για αντικείμενα ή πληροφορίες για τα χαρακτηριστικά των αντικειμένων. Είναι μία απλή και εύκολη τεχνική, η οποία βασίζεται μόνο σε λίγες παρατηρήσεις. Από την άλλη, το βασικό μειονέκτημα της τεχνικής είναι τα θέματα ασφάλειας και ιδιωτικότητας, που μπορεί να προκύψουν, καθώς και το γεγονός ότι η συλλογή των χαρακτηριστικών των χρηστών είναι πολλές φορές δύσκολη ή μη εφικτή [4].

Μία ακόμα κατηγορία συστημάτων είναι τα *Συστήματα Συστάσεων με Βάση την Κοινότητα* (Community-based Systems). Τα συστήματα αυτά παράγουν συστάσεις σε έναν χρήστη βασίζόμενα στις προτιμήσεις των «φίλων» του, των άλλων χρηστών δηλαδή με τους οποίους συνδέεται. Σε αντίθεση με τα συστήματα, τα οποία υλοποιούν την τεχνική του συνεργατικού φιλτραρίσματος και βασίζονται στην ομοιότητα των άλλων χρηστών, τα community-based συστήματα βασίζονται στην ομοιότητα μεταξύ των φίλων του χρήστη. Με άλλα λόγια, η λογική πίσω από τη συγκεκριμένη τεχνική είναι το γεγονός ότι οι άνθρωποι λαμβάνουν υπόψη περισσότερο τις προτάσεις των φίλων τους παρά τις συστάσεις τυχαίων ατόμων.

2.4 Αξιολόγηση Συστημάτων Συστάσεων

Μετά τη σχεδίαση και την υλοποίηση ενός συστήματος συστάσεων, το οποίο χρησιμοποιεί μία από τις προηγούμενες τεχνικές που περιγράφηκαν στην προηγούμενη ενότητα, είναι χρήσιμο να μπορεί να πραγματοποιηθεί μία αξιολόγησή του, προκειμένου να διαπιστωθεί ότι δουλεύει σωστά. Δεδομένου ότι τα

συστήματα συστάσεων συνήθως χρησιμοποιούνται για την επίτευξη ενός στόχου, είτε αυτός είναι η μεγιστοποίηση της ικανοποίησης των χρηστών ή η αύξηση των πωλήσεων, η δοκιμή αυτών των συστημάτων είναι απαραίτητη. Ωστόσο, η δοκιμή ενός αλγορίθμου σε ένα σύστημα, το οποίο λειτουργεί με πραγματικά δεδομένα χρηστών, είναι συνήθως δύσκολο και αρκετά δαπανηρό. Επομένως, η δοκιμή των αλγορίθμων προτού αυτοί χρησιμοποιηθούν σε πραγματικά συστήματα (offline algorithmic evaluation), είναι κάτι το οποίο συνήθως γίνεται, χρησιμοποιώντας ορισμένα σύνολα δεδομένων.

Η διαδικασία με την οποία γίνεται αυτή η “offline” αξιολόγηση, βασίζεται στον χωρισμό των δεδομένων σε δύο μέρη, ένα μέρος για εκπαίδευση (training set) και ένα μέρος για έλεγχο (testing set), όπως γίνεται και στη μηχανική μάθηση. Τα δεδομένα, τα οποία χωρίζονται, αποτελούν συλλογές βαθμολογιών χρηστών για διάφορα αντικείμενα, συνήθως με επιπλέον πληροφορίες (χαρακτηριστικά) για αυτά. Από τα δεδομένα, τα οποία χωρίστηκαν για εκπαίδευση, κατασκευάζεται το μοντέλο συστάσεων [12, 13]. Έπειτα, οι βαθμολογίες των χρηστών των δεδομένων ελέγχου, χωρίζονται και αυτές σε δύο μέρη. Το ένα μέρος (query set) τροφοδοτείται στο σύστημα ως οι έως τώρα δοθείσες βαθμολογίες των χρηστών, έτσι ώστε να γίνει πρόβλεψη των βαθμολογιών των αντικειμένων του άλλου μέρους (target set), ώστε να διαπιστωθεί η «απόσταση» των προβλέψεων από τα πραγματικά δεδομένα. Η διαδικασία αυτή συνήθως γίνεται επαναληπτικά k φορές χωρίζοντας τα δεδομένα σε k ίσα μέρη και χρησιμοποιώντας κάθε φορά ένα μέρος για έλεγχο και τα υπόλοιπα μέρη για εκπαίδευση (k -fold cross validation). Τα αποτελέσματα σε κάθε επανάληψη συμψηφίζονται και προκύπτει η συνολική απόδοση του συστήματος.

Το είδος της μετρικής που θα χρησιμοποιηθεί, εξαρτάται από διάφορους παράγοντες, όπως το είδος της τεχνικής που χρησιμοποιεί το σύστημα, τα χαρακτηριστικά των δεδομένων και το στόχο του συστήματος συστάσεων. Μία από αυτές τις μετρικές, είναι η ακρίβεια των συστημάτων, η οποία ορίζεται ως ο λόγος των προβλέψεων των προτιμήσεων προς τις πραγματικές προτιμήσεις των χρηστών. Οι μετρικές για την μέτρηση της ακρίβειας χωρίζονται σε δύο κατηγορίες, τις *Στατιστικές Μετρικές Ακρίβειας* (Statistical Accuracy Metrics) και στις *Μετρικές Ακρίβειας Υποστήριξης Απόφασης* (Decision Support Accuracy Metrics) [8].

2.4.1 Στατιστικές Μετρικές Ακρίβειας

Οι στατιστικές μετρικές ακρίβειας αξιολογούν την ακρίβεια, ενός συστήματος παραγωγής συστάσεων, συγκρίνοντας τις βαθμολογίες που έχουν προβλεφτεί με τις πραγματικές βαθμολογίες των χρηστών. Η **Μέση Απόλυτη Απόκλιση** (Mean Absolute Error - MAE) είναι η καθιερωμένη και η πιο δημοφιλής μετρική, η οποία χρησιμοποιείται για τη μέτρηση της απόκλισης των προβλέψεων από τις πραγματικές τιμές [2, 8, 12]. Η μέθοδος αυτή, υπολογίζει τη μέση τιμή των απόλυτων τιμών των διαφορών μεταξύ κάθε πρόβλεψης και πραγματικής βαθμολογίας για όλες τις βαθμολογίες των χρηστών στα δεδομένα ελέγχου και δίνεται από τον ακόλουθο τύπο:

$$MAE = \frac{1}{N} \sum_{u,i} |p_{u,i} - r_{u,i}|$$

όπου $p_{u,i}$ είναι η πρόβλεψη της βαθμολογίας του χρήστη u για το αντικείμενο i , $r_{u,i}$ είναι η πραγματική βαθμολογία και N ο αριθμός των βαθμολογιών στα δεδομένα ελέγχου. Όσο χαμηλότερη είναι η τιμή της μέσης απόλυτης απόκλισης τόσο πιο ακριβείς είναι οι συστάσεις του συστήματος. Για παράδειγμα, σε δεδομένα βαθμολογιών κλίμακας 1-5, μία τιμή μέσης απόλυτης απόκλισης ίση με 0,3, σηματοδοτεί μία μικρή απόκλιση κατά 0,3 της τιμής της πρόβλεψης από την πραγματική βαθμολογία. Το βασικό πρόβλημα αυτής της μετρικής είναι ότι οι τιμές των αποτελεσμάτων είναι δύσκολο να συγκριθούν με δεδομένα βαθμολογιών με άλλη κλίμακα.

Η **Κανονικοποιημένη Μέση Απόλυτη Απόκλιση** (Normalized Mean Absolute Error - NMAE) είναι μία μετρική, η οποία χρησιμοποιείται σε αυτές τις περιπτώσεις. Η κανονικοποιημένη μέση απόλυτη απόκλιση κανονικοποιεί τις αποκλίσεις διαιρώντας αυτές με μία περιοχή πιθανών βαθμολογιών, με αποτέλεσμα οι τελικές τιμές να βρίσκονται στην κλίμακα 0-1. Στον ακόλουθο τύπο υπολογισμού της μετρικής, οι όροι r_{high} και r_{low} εκφράζουν τα όρια της κλίμακας των βαθμολογιών των δεδομένων [12].

$$NMAE = \frac{1}{N(r_{high} - r_{low})} \sum_{u,i} |p_{u,i} - r_{u,i}|$$

Μία ακόμα μετρική, η οποία χρησιμοποιείται αρκετά, είναι η *Ρίζα Μέσης Τετραγωνικής Απόκλισης* (Root Mean Squared Error - RMSE). Η μετρική δίνει μεγαλύτερη έμφαση σε μεγαλύτερες αποκλίσεις και δίνεται από τον παρακάτω τύπο [8, 12]:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i} (p_{u,i} - r_{u,i})^2}$$

2.4.2 Μετρικές Ακρίβειας Υποστήριξης Απόφασης

Η δεύτερη κατηγορία μετρικών ακρίβειας των συστημάτων είναι οι μετρικές ακρίβειας υποστήριξης απόφασης. Σε αυτές, μεταξύ άλλων, ανήκουν η καμπύλη *Λειτουργικού Χαρακτηριστικού Δείκτη* (Receiver Operating Characteristic - ROC), *Καμπύλη Ακρίβειας Ανάκλησης* (Precision Recall Curve), η *Ακρίβεια* (Precision), η *Ανάκληση* (Recall) και η *F-μετρική* (F-measure). Οι μετρικές αυτές, θεωρούν την πρόβλεψη ως μία δυαδική διαδικασία, η οποία διαχωρίζει τα υψηλής ποιότητας (σύστασης) αντικείμενα από τα χαμηλής [8]. Η ακρίβεια μπορεί να οριστεί ως ο λόγος των σωστά προβλεφθέντων ως σχετικά (true positive) αντικειμένων προς όλα τα αντικείμενα που συστάθηκαν στον χρήστη (true positive + false positive) ενώ η ανάκληση είναι ο λόγος των σωστά προβλεφθέντων ως σχετικά (true positive) αντικειμένων προς όλα τα αντικείμενα τα οποία θεωρούνται σχετικά για τον χρήστη (true positive + false negative) [Εικόνα 2.5]. Αντίστοιχα με τις μετρικές ακρίβειας και ανάκλησης μπορούν να χρησιμοποιηθούν και οι κλασικές μετρικές, *Εναισθησία* (Sensitivity), η οποία είναι ισοδύναμη της ανάκλησης και *Εξειδίκευση* (Specificity), η οποία είναι ο λόγος των μη-σχετικών αντικειμένων που ορθά δεν προτάθηκαν στον χρήστη.

	Relevant	Irrelevant
Retrieved	TP	FP
Not retrieved	FN	TN

(a) Precision: $\frac{TP}{TP+FP}$

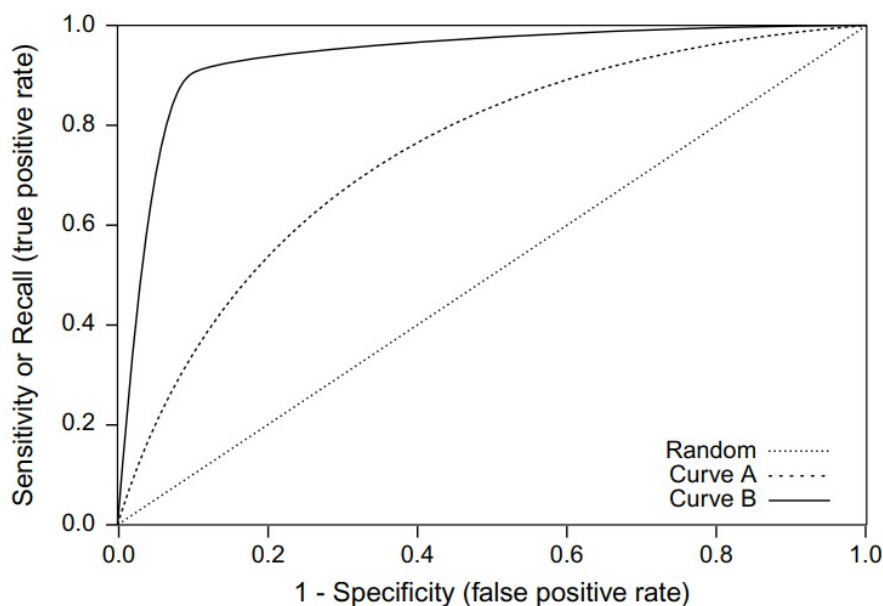
(b) Recall: $\frac{TP}{TP+FN}$

Εικόνα 2.5: Ακρίβεια και Ανάκληση στον Πίνακα Σύγχυσης [12]

Οι μετρικές της ακρίβειας και της ανάκλησης μπορούν να απλοποιηθούν σε μία μοναδική μετρική, την F-measure. Η μετρική συνδυάζει τις δύο μετρικές, δίνοντας την ίδια βαρύτητα. Το αποτέλεσμα κάνει την σύγκριση μεταξύ δύο αλγορίθμων και διαφορετικών δεδομένων εύκολη, ωστόσο χάνεται η πληροφορία για την τιμή των δύο μετρικών (π.χ. η μία μπορεί να είναι πολύ υψηλή και η άλλη πολύ χαμηλή, κάτι το οποίο μπορεί να μην είναι επιθυμητό) [12]. Η F-measure δίνεται υπολογίζεται ως εξής:

$$F - measure = \frac{2PR}{P+R}$$

Όπως προαναφέρθηκε, άλλη μία μετρική αυτής της κατηγορίας είναι η καμπύλη λειτουργικού χαρακτηριστικού δείκτη. Η μετρική αποτυπώνει την σχέση μεταξύ της ακρίβειας και της ανάκλησης, δείχνοντας την αύξηση της ανάκλησης ενός συστήματος καθώς μειώνονται οι απαιτήσεις για ακρίβεια. Με άλλα λόγια, η καμπύλη δείχνει ότι όσο προτείνονται σχετικά για τον χρήστη αντικείμενα, ανάμεσα από όλα τα σχετικά αντικείμενα, υπάρχει μείωση στην ακρίβεια καθώς τα σχετικά αντικείμενα τελειώνουν και αρχίζουν να προτείνονται τα λιγότερο σχετικά. Ένα σύστημα συστάσεων, το οποίο παράγει τυχαίες συστάσεις, αποτυπώνονται κατά μέσο όρο, ως μία διαγώνια ευθεία, ενώ καμπύλες που βρίσκονται πάνω από αυτή την ευθεία έχουν καλύτερη απόδοση [Εικόνα 2.6]. Όσο απομακρύνονται οι καμπύλες, προς τα πάνω αριστερά της, τόσο καλύτερο είναι το σύστημα συστάσεων [12].



Εικόνα 2.6: Παραδείγματα Καμπυλών ROC [12]

2.4.3 Άλλες Μετρικές Αξιολόγησης

Πέραν από την ακρίβεια, η οποία είναι μία πολύ σημαντική μετρική, υπάρχουν επίσης και άλλα κριτήρια, με τα οποία αξιολογούνται τα συστήματα συστάσεων. Ανάμεσα σε αυτά βρίσκεται η *Κάλυψη* (Coverage).

Η κάλυψη είναι μία μετρική που εκφράζει το ποσοστό των χρηστών, για τους οποίους ένα σύστημα συστάσεων μπορεί να παράξει συστάσεις [8]. Επίσης, σε παραλλαγή της, μπορεί να αποτυπώσει το ποσοστό των αντικειμένων που προτείνονται στους χρήστες. Η τιμή αυτής της μετρικής θα πρέπει να είναι πάντα υψηλή σε ένα σύστημα συστάσεων, καθώς μία υψηλότερη τιμή δείχνει ότι γίνονται συστάσεις για ένα μεγάλο μέρος του συνόλου των αντικειμένων [2]. Η παρακολούθησή της είναι πολύ σημαντική, καθώς διάφορες παραμετροποιήσεις, με σκοπό την αύξηση της απόδοσης του συστήματος, μπορεί να οδηγήσουν σε καταστάσεις όπου κάποια αντικείμενα έχουν σταματήσει να προτείνονται [9].

Ακόμα ένας τρόπος αξιολόγησης των συστημάτων συστάσεων είναι μέσω του κριτηρίου της *Πρωτοπορίας* (Novelty). Η πρωτοπορία είναι η ικανότητα των συστημάτων να προτείνουν αντικείμενα, για τα οποία δεν γνώριζε ο χρήστης. Πολλές εφαρμογές θεωρούν πολύ σημαντικό, τα αντικείμενα που προτείνονται στους χρήστες να είναι άγνωστα σε αυτούς, με χαρακτηριστικό παράδειγμα είναι οι εφαρμογές προβολής ειδήσεων, οι οποίες αντλούν άρθρα από πολλές πηγές. Η μέτρηση της πρωτοπορίας ενός συστήματος εμπεριέχει δυσκολίες, διότι απαιτεί την ανατροφοδότηση των χρηστών σχετικά με το αν γνώριζαν κάθε αντικείμενο που τους προτάθηκε [9].

Μία ακόμα μετρική αξιολόγησης συστημάτων συστάσεων είναι ο *Ρυθμός Εκπαίδευσης* (Learning Rate), η οποία μετράει το πόσο γρήγορα ένα σύστημα παράγει ποιοτικές συστάσεις από την στιγμή που θα αρχίσει να λαμβάνει δεδομένα. Η μετρική είναι ιδιαίτερα χρήσιμη, καθώς δείχνει τον αριθμό των βαθμολογιών που πρέπει να δώσει ο χρήστης ή τον αριθμό των αλληλεπιδράσεων που πρέπει να συλλεχθούν, προκειμένου το σύστημα να αρχίσει να παρέχει υψηλής ποιότητας συστάσεις.

Τέλος, δεδομένου ότι ένας από τους βασικούς στόχους των συστημάτων συστάσεων είναι η μεγιστοποίηση της ικανοποίησης των χρηστών, ένας τρόπος αξιολόγησής τους είναι η εύρεση του βαθμού στον οποίο τους ικανοποιούν. Αυτό μπορεί να επιτευχθεί διεξάγοντας μία μορφή έρευνας στους χρήστες μιας εφαρμογής και πραγματοποιώντας μία απλή στατιστική ανάλυση στα αποτελέσματα.

2.5 Προκλήσεις Συστημάτων Συστάσεων

Τα συστήματα συστάσεων έχουν ως γενικό σκοπό την παραγωγή ποιοτικών συστάσεων για τους χρήστες των εφαρμογών. Ωστόσο, στην προσπάθειά τους να επιτύχουν τον στόχο τους συναντούν διάφορες δυσκολίες και προκλήσεις, τις οποίες καλούνται να αντιμετωπίσουν. Τα προβλήματα αυτά μπορεί να προέρχονται από διάφορες αιτίες, όπως η χαμηλή διαθεσιμότητα δεδομένων εισόδου ή την ίδια την φύση των δεδομένων. Επίσης, αρκετά προβλήματα προκύπτουν αναφορικά με την προστασία της ιδιωτικότητας των χρηστών και την ασφάλεια του συστήματος απέναντι σε κακόβουλες ενέργειες. Τέλος, άλλα προβλήματα βρίσκονται κατά τη διάρκεια της διαδικασίας εκπαίδευσης του συστήματος, ενώ άλλα έχουν να κάνουν με την μείωση της αποτελεσματικότητάς των συστημάτων σε βάθος χρόνου.

2.5.1 Πρόβλημα Κρύας Εκκίνησης

Μία από τις βασικές προκλήσεις, την οποία καλούνται να υπερβούν τα συστήματα συστάσεων είναι το *Πρόβλημα Κρύας Εκκίνησης* (Cold-start Problem). Πρόκειται για μία κατάσταση κατά την οποία ένα σύστημα δεν έχει στη διάθεσή του επαρκεί αριθμό δεδομένων (ιστορικό βαθμολογιών, αλληλεπιδράσεων, χαρακτηριστικά) για έναν χρήστη ή για ένα αντικείμενο, προκειμένου να πραγματοποιήσει τις σχετικές προβλέψεις [2, 4, 5, 8]. Όταν ένας χρήστης ξεκινάει να χρησιμοποιεί το σύστημα, υπάρχουν λίγες πληροφορίες για τις προτιμήσεις του ενώ αντίστοιχα όταν προστίθεται για πρώτη φορά ένα νέο αντικείμενο στο σύστημα, δεν υπάρχουν βαθμολογίες ή δεδομένα αλληλεπιδράσεων διαθέσιμα για αυτό.

Το πρόβλημα κρύας εκκίνησης συναντάται κυρίως στα συστήματα που χρησιμοποιούν την τεχνική του συνεργατικού φιλτραρίσματος, όπως επίσης και όταν χρησιμοποιείται η τεχνική του φιλτραρίσματος με βάση το περιεχόμενο, ενώ μπορεί να αντιμετωπιστεί με διάφορους τρόπους. Ανάμεσα σε αυτούς τους τρόπους είναι να μπαίνει υποχρεωτικά ένας νέος χρήστης στη διαδικασία να βαθμολογεί κάποια αντικείμενα, να γίνονται συστάσεις στους νέους χρήστες με βάση τα δημογραφικά του χαρακτηριστικά [5], εάν αυτά είναι διαθέσιμα, όπως αντίστοιχα συμβαίνει στα συστήματα δημογραφικού φιλτραρίσματος. Τέλος, μία πιο απλή λύση στο πρόβλημα είναι η παραγωγή αρχικά τυχαίων συστάσεων στους νέους χρήστες και η επιτηδευμένη προώθηση των νέων αντικειμένων σε όλους τους χρήστες, έως ότου γίνει η συλλογή αρκετών δεδομένων. Από την άλλη, αυτή η λύση μπορεί να επηρεάσει αρνητικά την ακρίβεια του συστήματος.

2.5.2 Υπέρ-Εξειδίκευση

Η *Υπέρ-Εξειδίκευση* (Over-Specialisation) είναι ένα πρόβλημα των συστημάτων, το οποίο εντοπίζεται όταν οι χρήστες λαμβάνουν συστάσεις για αντικείμενα που ανήκουν αυστηρά στις προτιμήσεις τους και προέρχονται αποκλειστικά από κάποιες συγκεκριμένες κατηγορίες. Χαρακτηριστικό παράδειγμα αποτελεί ένα σύστημα, το οποίο προτείνει σε έναν χρήστη βιβλία ενός συγκεκριμένου συγγραφέα, για τον λόγο ότι ο χρήστης αυτός αγοράζει μόνο τέτοια βιβλία. Αυτό έχει ως αποτέλεσμα να μην υπάρχει ποικιλία στα αντικείμενα που προτείνονται και άρα ο χρήστης να μην ανακαλύπτει νέο περιεχόμενο, το οποίο μπορεί να του αρέσει και εν τέλη να αποδειχθεί ωφέλιμο [2, 4]. Το πρόβλημα της υπερ-εξειδίκευσης των συστάσεων, συναντάται συνήθως στα συστήματα που χρησιμοποιούν την τεχνική φιλτραρίσματος με βάση το περιεχόμενο.

Προκειμένου να προτείνονται πρωτότυπα αντικείμενα, ανάμεσα σε άλλα γνώριμα για τον χρήστη και άρα να αποφευχθεί το συγκεκριμένο πρόβλημα, είναι απαραίτητο να εισαχθεί τυχαιότητα στην παραγωγή των συστάσεων. Αυτό μπορεί να γίνει χρησιμοποιώντας γενετικούς αλγόριθμους, οι οποίοι εξάγουν ποικίλες συστάσεις [4].

Από την άλλη, τα συστήματα συστάσεων συνεργατικού φιλτραρίσματος, δεν εμφανίζουν αυτό το πρόβλημα σε μεγάλο βαθμό, καθώς είναι πιθανό να προτείνονται διάφορων ειδών αντικείμενα.

2.5.3 Κλιμακωσιμότητα

Άλλη μία πρόκληση για τα συστήματα συστάσεων είναι ο συνεχώς αυξανόμενος όγκος δεδομένων εισόδου, ο οποίος προκύπτει σε πραγματικές εφαρμογές. Το πρόβλημα της *Κλιμακωσιμότητας* (“Scalability”) είναι ένα πολύ συνηθισμένο πρόβλημα, διότι οι απαιτούμενοι υπολογιστικοί πόροι συνήθως αυξάνονται γραμμικά σε σχέση με την αύξηση του αριθμού των χρηστών και των αντικειμένων. Οι αλγόριθμοι που είναι αποτελεσματικοί, έχοντας έναν ορισμένο αριθμό δεδομένων εισόδου, είναι πιθανό να μην παράγουν ικανοποιητικές συστάσεις όταν ο όγκος της πληροφορίας βρίσκεται εκτός αυτών των ορίων [2, 4].

Για την αντιμετώπιση του προβλήματος είναι απαραίτητο να χρησιμοποιηθούν τεχνικές, οι οποίες είναι σε θέση να κλιμακώνονται, καθώς αυξάνεται ο αριθμός των δεδομένων, ώστε να παραμένει υψηλή η ποιότητα των συστάσεων του συστήματος. Ορισμένες από αυτές τις τεχνικές είναι η χρήση τεχνικών συσταδοποίησης [5] ή μείωσης των διαστάσεων, όπως η μέθοδος Αποσύνθεσης Ιδιαζουσών Τιμών [8].

2.5.4 Αραιότητα Δεδομένων

Στην περίπτωση μίας εφαρμογής, η οποία περιλαμβάνει έναν μεγάλο αριθμό αντικειμένων, οι χρήστες είναι πιθανό να «καταναλώνουν» ή να βαθμολογούν ένα μικρό ποσοστό αυτών. Το σύστημα συστάσεων μιας τέτοιας εφαρμογής καλείται να παράξει συστάσεις χρησιμοποιώντας αυτά τα ελάχιστα δεδομένα, τα οποία συνήθως είναι διασκορπισμένα και αραιά [2]. Το πρόβλημα της *Αραιότητας* (Sparsity) των δεδομένων, όπως αποκαλείται, οδηγεί στην ύπαρξη ενός αραιού πίνακα χρηστών-αντικειμένων, στην αποτυχία εντοπισμού γειτονικών χρηστών ή αντικειμένων και εν τέλει στην παραγωγή συστάσεων μειωμένης ακρίβειας αλλά και σε προβλήματα χαμηλής κάλυψης [8].

Το πρόβλημα εντοπίζεται τόσο σε συστήματα, τα οποία χρησιμοποιούν την τεχνική του συνεργατικού φιλτραρίσματος, όσο και σε άλλα που κάνουν χρήση άλλων τεχνικών, με εξαίρεση τα συστήματα φιλτραρίσματος με βάση το περιεχόμενο. Τα συστήματα αυτά δεν επηρεάζονται καθώς οι συστάσεις τους βασίζονται στα χαρακτηριστικά των χρηστών και των αντικειμένων.

Επομένως, μία λύση στο πρόβλημα δίνεται χρησιμοποιώντας την τεχνική του φιλτραρίσματος με βάση το περιεχόμενο, ως αποκλειστική τεχνική ή σε συνδυασμό με κάποια άλλη (υβριδικό φιλτράρισμα). Επίσης, η χρήση τεχνικών δημογραφικού φιλτραρίσματος και τεχνικών μείωσης των διαστάσεων, όπως η μέθοδος SVD μπορούν επίσης να αποδειχθούν αποτελεσματικές [5].

2.5.5 Ιδιωτικότητα

Η διασφάλιση της *Ιδιωτικότητας* (Privacy) των χρηστών είναι μία ακόμα πρόκληση, την οποία τα συστήματα συστάσεων πρέπει να υπερβούν. Συνήθως, ένα σύστημα συστάσεων συλλέγει διαρκώς όποια δεδομένα είναι διαθέσιμα για τους χρήστες (είτε τα παρέχουν οι ίδιοι είτε όχι) και τα αντικείμενα, με σκοπό να βελτιώσει την ποιότητα των συστάσεων του. Με άλλα λόγια, όσο περισσότερα δεδομένα συλλέγονται για τους χρήστες, τόσο πιο πιθανό είναι να γίνεται μία σωστή πρόβλεψη των προτιμήσεών τους και να λαμβάνουν καλύτερες συστάσεις [2, 9]. Ωστόσο, το γεγονός αυτό επηρεάζει αρνητικά την ιδιωτικότητα των χρηστών, καθώς το σύστημα αρχίζει να γνωρίζει περισσότερες πληροφορίες για τον χρήστη από όσες απαιτούνται. Επίσης, δημιουργούνται σημαντικά προβλήματα ιδιωτικότητας στην περίπτωση όπου συμβεί μία ανεπιθύμητη *αποκάλυψη* (exposure) αυτών των δεδομένων, καθώς καταρρίπτεται η ανωνυμία τους.

Η πρόκληση της διασφάλισης της ιδιωτικότητας των χρηστών συναντάται σε όλα τα συστήματα συστάσεων ανεξάρτητα από την τεχνική, την οποία χρησιμοποιούν, αλλά σε διαφορετικό βαθμό στο κάθε ένα. Τα συστήματα δημογραφικού φιλτραρίσματος, για παράδειγμα, τα οποία συλλέγουν δεδομένα όπως το φύλο, η ηλικία και η τοποθεσία των χρηστών, έχουν μεγαλύτερο αντίκτυπο όσον αφορά την ιδιωτικότητα σε περίπτωση μιας ενδεχόμενης διαρροής [4]. Αντίστοιχα, το πρόβλημα υπάρχει και στα συστήματα συνεργατικού φιλτραρίσματος και φιλτραρίσματος με βάση το περιεχόμενο. Ωστόσο, για τον λόγο ότι τα πρώτα χρησιμοποιούν τα δεδομένα των άλλων χρηστών για να παράξουν συστάσεις, τα κάνει περισσότερο επιρρεπή σε θέματα ιδιωτικότητας [5]. Επίσης, τα συστήματα συνεργατικού φιλτραρίσματος, απαιτούν σε

μεγαλύτερο βαθμό δεδομένα προσωπικών προτιμήσεων από τον χρήστη, οδηγώντας σε μεγαλύτερους προβληματισμούς όσον αφορά την ιδιωτικότητα, καθώς τα δεδομένα αυτά μπορούν να αποκαλύψουν σημαντικές προσωπικές πληροφορίες του χρήστη [14].

Η παροχή περισσότερων δεδομένων εισόδου σε ένα σύστημα συνεργατικού φιλτραρίσματος, αυξάνει την ακρίβεια των συστάσεων του, αλλά επίσης αυξάνεται και ο κίνδυνος μίας μη επιθυμητής αποκάλυψης προσωπικών πληροφοριών. Το ιδανικό θα ήταν, να υπάρχει ένα όριο στη συλλογή των δεδομένων, στο οποίο το σύστημα είναι σε θέση να παράγει ποιοτικές συστάσεις χωρίς όμως να απαιτείται η συλλογή πάρα πολλών δεδομένων για τους χρήστες. Για να επιτευχθεί ωστόσο κάτι τέτοιο, προκύπτουν, μεταξύ άλλων, ερωτήματα σχετικά με το ποια δεδομένα θα πρέπει να διατηρεί το σύστημα, ποια είναι η ιδανική ποσότητα αυτών των δεδομένων και ποιο είναι το αντίκτυπο στους αλγόριθμους συνεργατικού φιλτραρίσματος [14].

Τα δεδομένα προτιμήσεων, τα οποία συλλέγονται, βοηθούν το σύστημα να διακρίνει έναν χρήστη ανάμεσα σε άλλους. Έτσι ορισμένα δεδομένα συμβάλουν περισσότερο από κάποια άλλα στην προσπάθεια αυτή και άρα θεωρούνται ότι έχουν μεγαλύτερη αξία. Από την άλλη, τα δεδομένα αυτά, λόγω του ότι μπορούν να προσδιορίσουν με μεγαλύτερη ευκολία τον χρήστη ενισχύουν το πρόβλημα της διασφάλισης της ιδιωτικότητας. Επομένως, έχοντας μία μετρική, η οποία εκφράζει την αξία των δεδομένων, μπορεί οριστεί το σημείο που θα πρέπει να σταματάει η συλλογή τους, διασφαλίζοντας έτσι την ιδιωτικότητα των χρηστών και παράλληλα την ποιότητα των συστάσεων. Παρόλα αυτά, μειώνοντας την ποσότητα των δεδομένων, επηρεάζεται η αποτελεσματικότητα των αλγορίθμων συνεργατικού φιλτραρίσματος, όσον αφορά την ακρίβεια, την κάλυψη και την πρωτοπορία των παραγόμενων συστάσεων.

Τέλος, αξίζει να σημειωθεί ότι συστήματα με καταναεμημένες αρχιτεκτονικές, έχουν μεγαλύτερες πιθανότητες στο να γίνει μία αποκάλυψη των δεδομένων των χρηστών. Αυτό ωστόσο μπορεί να διασφαλιστεί χρησιμοποιώντας συμμετρική κρυπτογράφηση και τεχνικές δημοσίου κλειδιού στις επικοινωνίες [9].

2.5.6 Επιθέσεις Κακόβουλης Τροφοδότησης (Shilling Attacks)

Οι *Επιθέσεις Κακόβουλης Τροφοδότησης* (Shilling Attacks) είναι μία ακόμα πρόκληση, με την οποία ενδεχομένως να βρεθεί αντιμέτωπο ένα σύστημα συστάσεων. Οι επιθέσεις αυτές, μπορούν να μειώσουν την αξιοπιστία του συστήματος, καθώς και να μειώσουν την απόδοσή και την ποιότητα των παραγόμενων συστάσεων. Πιο συγκεκριμένα, οι επιθέσεις κακόβουλης τροφοδότησης έχουν σκοπό την χειραγώγηση των συστάσεων ενός συστήματος και διακρίνονται σε δύο κατηγορίες, τις *επιθέσεις με σκοπό την προώθηση* (push attacks) και τις *επιθέσεις με σκοπό την υπονόμευση* (nuke attacks) [2]. Οι μεν πρώτες γίνονται από έναν χρήστη με σκοπό την προώθηση ενός αντικειμένου τροφοδοτώντας το σύστημα με καλές αξιολογήσεις για αυτό. Παράδειγμα ενός τέτοιου χρήστη μπορεί να αποτελέσει ένας κατασκευαστής ενός προϊόντος, ο οποίος επιθυμεί να αυξήσει τις πωλήσεις του μέσω μιας εφαρμογής ηλεκτρονικών αγορών. Από την άλλη, οι δε δεύτερες επιθέσεις έχουν ως σκοπό τη δυσφήμιση ενός αντικειμένου παρέχοντας στο σύστημα αρνητικές βαθμολογίες για αυτό. Σε αυτή την περίπτωση, με βάση το προηγούμενο παράδειγμα, ο κατασκευαστής εδώ επιχειρεί να αυξήσει τις πωλήσεις του υπονομεύοντας την φήμη των προϊόντων των ανταγωνιστών.

Από τις επιθέσεις κακόβουλης τροφοδότησης απειλούνται κυρίως τα συστήματα συνεργατικού φιλτραρίσματος, καθώς αυτά βασίζονται αποκλειστικά σε δεδομένα βαθμολογιών των χρηστών για αντικείμενα, για την εξαγωγή συστάσεων [5].

Μία επίθεση κακόβουλης τροφοδότησης μπορεί να επιτευχθεί έχοντας μία ομάδα χρηστών (πραγματικοί ή τεχνητοί), η οποία παρέχει στο σύστημα βαθμολογίες που κατευθύνουν το σύστημα στην επιθυμητή συμπεριφορά. Το είδος των επιθέσεων αυτών καθορίζει τον τρόπο με τον οποίο θα παραχθούν αυτές οι βαθμολογίες, όσο και τον βαθμό της επιτυχίας της επίθεσης με βάση τον αλγόριθμο συνεργατικού φιλτραρίσματος που χρησιμοποιεί το σύστημα [14].

Οι shilling επιθέσεις μπορούν να αντιμετωπιστούν με διάφορους τρόπους, όπως με την χρήση αλγορίθμων, οι οποίοι είναι ανθεκτικοί σε αυτές τις επιθέσεις, ή αυξάνοντας το βαθμό δυσκολίας απόκτησης ενός λογαριασμού χρήστη, σε επίπεδο εφαρμογής (π.χ. χρησιμοποιώντας CAPTCHAs). Έχει αποδειχθεί ότι οι

αλγόριθμοι συνεργατικού φιλτραρίσματος, οι οποίοι βασίζονται σε έμμεσα δεδομένα (ιστορικό αλληλεπιδράσεων χρηστών με αντικείμενα), καθιστούν τις επιθέσεις λιγότερο αποτελεσματικές [14].

2.5.7 Αδυναμία Εξέλιξης Σύμφωνα με το Περιεχόμενο

Τέλος, μία βασική πρόκληση των συστημάτων συστάσεων είναι η αδυναμία ενημέρωσης των προτάσεων τους και εξέλιξή τους με βάση το περιεχόμενο. Τα συστήματα θα πρέπει να είναι σε θέση να προσαρμόζονται γρήγορα στις αλλαγές των δεδομένων των εφαρμογών. Για παράδειγμα, όταν προστίθενται νέα δεδομένα βαθμολογιών, ή εισάγονται νέα αντικείμενα, θα πρέπει να αυξάνεται το εύρος των πιθανών συστάσεων [15]. Η συνεχή ενημέρωση του συστήματος συμβάλει στο να διατηρείται σε υψηλά επίπεδα η ακρίβειά του, καθώς συνεχίζονται να παράγονται ποιοτικές συστάσεις στους χρήστες, πάρα τις όποιες ενδεχόμενες αλλαγές στις προτιμήσεις τους ή του περιεχομένου της εφαρμογής.

Λύση στο πρόβλημα αυτό, δίνει η εξ αρχής σχεδίαση των συστημάτων συστάσεων, ώστε να λειτουργούν χρησιμοποιώντας δεδομένα εισόδου με την μορφή συνεχών ροών δεδομένων. Παράλληλα, πρέπει να χρησιμοποιούνται οι κατάλληλες τεχνικές και αλγόριθμοι, οι οποίοι υποστηρίζουν την ανάλυση αυτών των δεδομένων.

2.6 Συνεχούς Ροής Δεδομένα

Η *ροή δεδομένων* (data stream) είναι μία αφαιρετική έννοια αναφερόμενη σε ένα *άνευ ορίων σετ δεδομένων* (unbounded dataset), το οποίο είναι συνεχώς αυξανόμενο [16]. Το σετ δεδομένων αυτό δεν έχει όρια καθώς νέες εγγραφές – γεγονότα προκύπτουν κάθε χρονική στιγμή, ενώ συνήθως αναφέρεται και ως *ροή γεγονότων* (event stream) ή δεδομένα συνεχούς ροής. Χαρακτηριστικά παραδείγματα, αυτών των διαρκώς παραγόμενων δεδομένων, είναι οι τιμές αισθητήρων, οι συναλλαγές μία πιστωτικής κάρτας, οι κινήσεις των χρηστών σε μία ιστοσελίδα, οι τιμές ενός χρηματιστηριακού δείκτη και οι κινήσεις των δεδομένων σε ένα δίκτυο.

Πέρα από το γεγονός ότι τα συνεχούς ροής δεδομένα δεν έχουν όρια ως προς την έκτασή τους, υπάρχουν και άλλα χαρακτηριστικά, τα οποία τα περιγράφουν. Ανάμεσα σε αυτά, είναι η προκαθορισμένη σειρά των γεγονότων, η αμετάβλητη φύση τους και η δυνατότητα επανάληψής τους. Πιο συγκεκριμένα, όσον αφορά την σειρά των γεγονότων, είναι κάτι το οποίο χαρακτηρίζει αυτά τα δεδομένα ενώ παίζει σημαντικό ρόλο για την ερμηνεία τους. Αναφορικά με το γεγονός ότι δεν μεταβάλλονται, ένα καλό παράδειγμα αποτελούν τα δεδομένα χρηματικών συναλλαγών. Οι συναλλαγές, οι οποίες ακυρώνονται, δεν παύουν να υπάρχουν αλλά αναιρούνται μέσω νέων γεγονότων. Τέλος, οι ροές δεδομένων πρέπει να έχουν τη δυνατότητα να επαναλαμβάνονται, προκειμένου να μπορούν να αναλύονται εκ νέου με άλλες μεθόδους ή να πραγματοποιούνται έλεγχοι.

Γενικότερα, τα συνεχούς ροής δεδομένα χρησιμοποιούνται με σκοπό την εξαγωγή *γνώσεων* (insights) ή τον έλεγχο εφαρμογών. Ωστόσο, αυτές οι διαδικασίες απαιτούν να έχει προηγηθεί η επεξεργασία αυτών των δεδομένων και η ανάλυσή τους.

2.6.1 Επεξεργασία Ροών Δεδομένων

Η Επεξεργασία Συνεχούς Ροής Δεδομένων (Stream Processing) είναι μία συνεχής επεξεργασία των δεδομένων καθώς αυτά παράγονται ή λαμβάνονται [17]. Σε πολλές εφαρμογές, όπως τις εφαρμογές ανίχνευσης απατών σε πιστωτικές κάρτες, τα ιατρικά συστήματα με αισθητήρες, τις IoT εφαρμογές, τα συστήματα συστάσεων πραγματικού χρόνου, η επεξεργασία των δεδομένων πρέπει να γίνεται τη στιγμή της παραγωγής τους. Σε αντίθετη περίπτωση, οι εφαρμογές αυτές αποκλίνουν από το στόχο τους ή καθίστανται λιγότερο αποτελεσματικές.

Η επεξεργασία συνεχούς ροής δεδομένων μπορεί να περιλαμβάνει την πραγματοποίηση διάφορων «εργασιών» πάνω στα δεδομένα, οι οποίες μεταξύ άλλων μπορεί να είναι οι *συναθροίσεις* (aggregations)

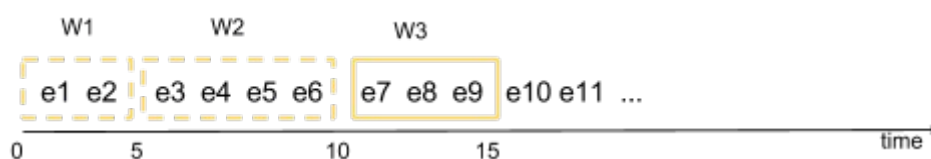
(π.χ. άθροιση, εύρεση μέσου όρου), η ανάλυση (π.χ. εύρεση μοτίβων στα δεδομένα προκειμένου να γίνεται πρόβλεψη των μελλοντικών τιμών) και οι *μετασχηματισμοί* (transformations) (π.χ. αλλαγή τύπων δεδομένων). Οι εργασίες αυτές μπορεί να γίνονται σειριακά, παράλληλα ή και με συνδυασμό των δύο τρόπων. Η σειρά των εργασιών αυτών αναφέρεται ως *pipeline επεξεργασίας ροών δεδομένων* (stream processing pipeline) και περιλαμβάνει την παραγωγή των δεδομένων, την επεξεργασία τους και την μεταφορά τους στον τελικό προορισμό [18].

Πρέπει επίσης να αναφερθεί, ότι υπάρχουν εξειδικευμένα εργαλεία, τα οποία βοηθούν στην επεξεργασία των συνεχούς ροής δεδομένων και αναφέρονται ως *Μηχανές Επεξεργασίας Ροών Δεδομένων* (Stream Processing Engines). Μία μηχανή επεξεργασίας ροών δεδομένων περιλαμβάνει βιβλιοθήκες με κώδικα, ο οποίος έχει ως σκοπό την επεξεργασία αυτού του είδους των δεδομένων. Συνήθως, τα περισσότερα εργαλεία αυτού του είδους έχουν κάποιες βασικές λειτουργίες. Ανάμεσα σε αυτές είναι τα *Παράθυρα Επεξεργασίας* (Windowing), η *Διαχείριση Κατάστασης* (State Management), η *Εγγύηση Επεξεργασίας Μηνύματος* (Message Processing Guarantee) και η *Ανθεκτικότητα σε Σφάλματα* (Fault Tolerance) [17].

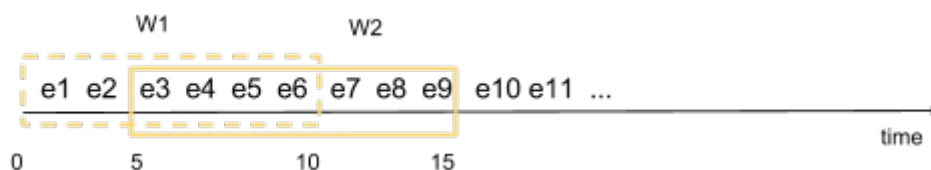
2.6.2 Βασικές Λειτουργίες των Stream Processing Engines

Όπως αναφέρθηκε και προηγουμένως, μία από τις βασικές λειτουργίες των μηχανών επεξεργασίας ροών δεδομένων είναι το “**windowing**”. Η λογική πίσω από αυτή τη λειτουργία είναι ο χωρισμός μίας άπειρης ροής δεδομένων σε πεπερασμένου μήκους τμήματα (παράθυρα γεγονότων). Ένα παράθυρο μπορεί να οριστεί με βάση τη *χρονική διάρκεια άφιξης των γεγονότων* (event-time), τη *χρονική διάρκεια επεξεργασίας των γεγονότων* (processing-time) ή με βάση τον αριθμό των γεγονότων [17]. Έχοντας για παράδειγμα ορίσει την τμηματοποίηση να γίνεται με βάση ένα χρονικό διάστημα, τα γεγονότα επεξεργάζονται, εξάγεται το αποτέλεσμα και τέλος απορρίπτονται με το πέρας του χρονικού ορίου. Αντίστοιχα, στην τμηματοποίηση με βάση τον αριθμό των γεγονότων, γίνεται η επεξεργασία τους, εξάγεται το αποτέλεσμα της επεξεργασίας και απορρίπτονται αφού έχει συμπληρωθεί ο αριθμός γεγονότων του κάθε παραθύρου.

Τα παράθυρα γεγονότων, είτε έχουν όρια με βάση το χρόνο είτε με βάση τον αριθμό των εγγραφών, μπορούν να είναι *σταθερά* (fixed windows) ή *κυλιόμενα* (sliding windows). Τα σταθερά παράθυρα έχουν σταθερό μήκος και το κάθε γεγονός ανήκει μόνο σε ένα παράθυρο [Εικόνα 2.7]. Από την άλλη τα κυλιόμενα παράθυρα έχουν επίσης σταθερό μήκος αλλά χωρίζονται από ένα ορισμένο μεσοδιάστημα χρόνου ή αριθμού γεγονότων αντίστοιχα [19]. Στην περίπτωση, κατά την οποία το μεσοδιάστημα αυτό είναι μικρότερο από το μήκος του παραθύρου, προκύπτει ότι κάποια γεγονότα θα ανήκουν σε περισσότερα από ένα παράθυρα [Εικόνα 2.8]. Είναι προφανές ότι όταν το μεσοδιάστημα είναι ίσο με το μήκος των κυλιόμενων παραθύρων, τότε τα παράθυρα γίνονται σταθερά [17].



Εικόνα 2.7: Σταθερά Παράθυρα Μήκους 5 Δευτερολέπτων [19]



Εικόνα 2.8: Κυλιόμενα Παράθυρα Μήκους 10 και Μεσοδιάστημα 5 Δευτερολέπτων [19]

Άλλη πολύ σημαντική λειτουργία των εργαλείων επεξεργασίας ροών δεδομένων είναι **διαχείριση κατάστασης**. Σε ορισμένα είδη επεξεργασίας, όπως η καταμέτρηση στοιχείων, είναι απαραίτητο να διατηρείται η κατάσταση τους. Επίσης, άλλο βασικό χαρακτηριστικό τους είναι η **εγγύηση επεξεργασίας**

μηνύματος, η οποία μπορεί να οριστεί σε τρία επίπεδα. Στο πρώτο επίπεδο δίνεται εγγύηση ότι κάθε γεγονός θα επεξεργάζεται το πολύ μία φορά, το οποίο σημαίνει ότι μπορεί κάποιο να χαθεί. Στο δεύτερο επίπεδο τα γεγονότα επεξεργάζεται τουλάχιστον μία φορά, χωρίς όμως να διασφαλίζεται ότι δεν θα συμπεριληφθούν ξανά, ενώ το τελευταίο επίπεδο εγγυάται ότι τα γεγονότα επεξεργάζονται ακριβώς μία φορά. Η επιλογή του επιπέδου εγγύησης, πρέπει να πραγματοποιείται ανάλογα με της ανάγκες κάθε εφαρμογής, καθώς γίνεται ένα αντιστάθμισμα αξιοπιστίας και κόστους. Τέλος, σημαντικό χαρακτηριστικό είναι και η ανθεκτικότητα σε σφάλματα. Πιο συγκεκριμένα, σε περίπτωση σφάλματος, το εργαλείο θα πρέπει να συνεχίζει την επεξεργασία από το σημείο της διακοπής, διότι η επεξεργασία των δεδομένων από την αρχή καθίσταται αδύνατη για πρακτικούς λόγους [17].

2.6.3 Επεξεργασία Μικρο-Ομάδων

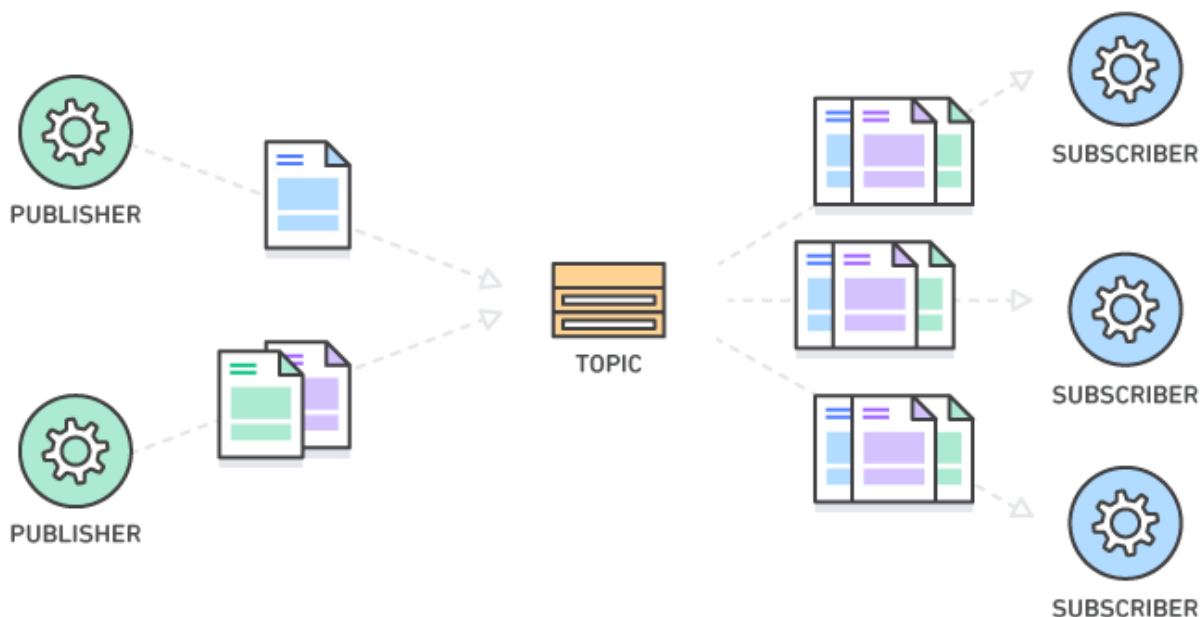
Παρόλα αυτά, η επεξεργασία των δεδομένων ακριβώς τη στιγμή που λαμβάνονται και η ενημέρωση του αποτελέσματος σε πραγματικό χρόνο, δεν είναι επιτακτική ανάγκη για όλες τις εφαρμογές [20]. Η λειτουργία ορισμένων εφαρμογών, δεν επηρεάζεται σε μεγάλο βαθμό από μία σχεδόν πραγματικού χρόνου επεξεργασία των δεδομένων τους, έτσι ώστε να χάνεται η αποτελεσματικότητά τους. Για να επιτευχθεί μία τέτοιου είδους επεξεργασία, συνήθως χρησιμοποιείται η Επεξεργασία Μικρο-Ομάδων (Micro-Batch Processing).

Η επεξεργασία μικρο-ομάδων, όπως υποδεικνύει το όνομά της, χωρίζει τα δεδομένα σε μικρές ομάδες, οι οποίες επεξεργάζονται σε μεγάλη συχνότητα. Επομένως, μία επεξεργασία ροών δεδομένων μπορεί να γίνει ως μία σειρά μετασχηματισμών σε πεπερασμένα σετ δεδομένων, χωρίζοντας τις σε μικρές ομάδες και ορίζοντας τη σειριακή επεξεργασία τους [21].

2.7 Ανταλλαγή Μηνυμάτων Publish / Subscribe

Αναφορικά με τη μεταφορά των δεδομένων συνεχούς ροής, μεταξύ άλλων, χρησιμοποιείται συνήθως το μοτίβο ανταλλαγής μηνυμάτων Publish / Subscribe. Με την βοήθεια αυτού του μοτίβου, τα γεγονότα μπορούν να μεταφέρονται από και προς όλα τα μέρη μιας εφαρμογής καθώς και να τροφοδοτούνται σε μία μηχανή επεξεργασίας ροών δεδομένων.

Η ανταλλαγή μηνυμάτων publish / subscribe είναι ενός είδους ασύγχρονη επικοινωνία *υπηρεσιών προς υπηρεσίες* (service-to-service), το οποίο χρησιμοποιείται σε *άνευ εξυπηρετητή* (serverless) αρχιτεκτονικές και αρχιτεκτονικές *μικρο-υπηρεσιών* (microservices) [22]. Σε αυτού του είδους την επικοινωνία, οι παραγωγοί μηνυμάτων (publishers) προωθούν τα μηνύματα σε ένα *θέμα* (topic) ενώ όσοι *καταναλωτές* (subscribers) ενδιαφέρονται για αυτό το θέμα, λαμβάνουν άμεσα τα μηνύματα. Ένας καταναλωτής μπορεί ταυτόχρονα να είναι και παραγωγός μηνυμάτων, ενώ επίσης δε γνωρίζει από ποιόν παραγωγό λαμβάνει τα δεδομένα ή για ποιους καταναλωτές προορίζονται αυτά που παράγει.



Εικόνα 2.9: Παράδειγμα Μοτίβου Ανταλλαγής Μηνυμάτων Publish / Subscribe [22]

2.8 Σύστημα Συστάσεων Πραγματικού Χρόνου

Έχοντας αναφερθεί στο γεγονός ότι τα περισσότερα συστήματα συστάσεων αδυνατούν να ενημερώνονται άμεσα σύμφωνα με τις αλλαγές στο περιεχόμενο των εφαρμογών, ως μία από τις βασικές προκλήσεις τους, διάφορα συστήματα έχουν σχεδιαστεί με πρωταρχικό στόχο την αντιμετώπιση της. Τα συστήματα συστάσεων *πραγματικού χρόνου* (real-time), όπως αυτά αποκαλούνται, αξιοποιούν ένα είδος δεδομένων εισόδου με τη μορφή συνεχών ροών και κάποιες από τις τεχνικές παραγωγής συστάσεων προκειμένου να παράγουν ακριβείς και ποιοτικές κάθε χρονική στιγμή συστάσεις.

Ένα τέτοιο σύστημα συστάσεων πραγματικού χρόνου είναι το “StreamRec” [15]. Το σύστημα αυτό χρησιμοποιεί τη μέθοδο του συνεργατικού φιλτραρίσματος με βάση την μνήμη και πιο συγκεκριμένα με βάση το αντικείμενο, ενώ ως είσοδο αξιοποιεί δεδομένα βαθμολογιών των χρηστών για αντικείμενα. Αναλυτικότερα, η είσοδος του συστήματος αποτελείται από δύο ειδών συνεχούς ροής γεγονότων, τα γεγονότα ενημέρωσης και τα γεγονότα σύστασης. Τα πρώτα, είναι στην πράξη οι βαθμολογίες των χρηστών, ενώ τα γεγονότα σύστασης είναι αιτήματα παραγωγής συστάσεων για έναν χρήστη. Η διαδικασία κατασκευής του μοντέλου και της παραγωγής συστάσεων, στο σύστημα αυτό, είναι μία σειρά μετασχηματισμών πάνω στα συνεχούς ροής δεδομένα.

Αντίστοιχα, το “ScaR” είναι ένα άλλο σύστημα συστάσεων, το οποίο στρέφεται γύρω από την παροχή ποιοτικών συστάσεων κάθε στιγμή [23]. Πιο συγκεκριμένα, το “ScaR” είναι ένα *πλαίσιο* (framework) βασισμένο στην Java, παραγωγής συστάσεων πραγματικού χρόνου. Βασικό του χαρακτηριστικό είναι ότι ακολουθεί μία αρχιτεκτονική μικρο-υπηρεσιών, όπου κάθε μία υπηρεσία είναι ανεξάρτητη και αναλαμβάνει ένα μέρος της διαδικασίας, ενώ η επικοινωνία μεταξύ τους επιτυγχάνεται μέσω μιας HTTP/REST διεπαφής.

Μία από αυτές τις μικρο-υπηρεσίες του συστήματος, είναι η «Υπηρεσία Επιπέδου Μετατροπής των Δεδομένων», η οποία διαχειρίζεται όλες τις επικοινωνίες με το “Apache Solr”, το οποίο είναι υπεύθυνο για λειτουργίες όπως την υποβολή ερωτημάτων ή την αποθήκευση των δεδομένων. Το σύστημα έχει σχεδιαστεί ώστε να μπορεί να δεχθεί διάφορους τύπους δεδομένων, όπως βαθμολογίες χρηστών, περιγραφές των αντικειμένων, δεδομένα από μέσα κοινωνικής δικτύωσης και δεδομένα για την τοποθεσία του χρήστη. Επίσης, μία άλλη υπηρεσία του “ScaR” είναι η «Υπηρεσία Μηχανής Συστάσεων», μέσω της οποίας

παράγονται οι συστάσεις με διάφορους τρόπους. Το σύστημα δίνει την επιλογή χρήσης διάφορων τεχνικών, όπως το συνεργατικό φιλτράρισμα με βάση τον χρήστη, φιλτράρισμα με βάση το περιεχόμενο, *Μέθοδο του πιο Δημοφιλούς* (Most Popular) (προτείνει τα πιο δημοφιλή αντικείμενα), ή ένα συνδυασμό αυτών (υβριδικό φιλτράρισμα). Οι αλγόριθμοι που χρησιμοποιούνται κάθε φορά μπορούν να παραμετροποιηθούν μέσω της «Υπηρεσίας Παραμετροποίησης του Συστήματος Συστάσεων». Για παράδειγμα, μέσω της υπηρεσίας μπορεί να οριστεί η μετρική ομοιότητας που θα χρησιμοποιηθεί, στην περίπτωση του συνεργατικού φιλτραρίσματος, ή η συνεισφορά του κάθε αλγορίθμου στο αποτέλεσμα, όταν γίνεται χρήση μιας υβριδικής τεχνικής. Επιπροσθέτως, υπάρχει η «Υπηρεσία Αξιολόγησης του Συστήματος Συστάσεων», η οποία δίνει την δυνατότητα να πραγματοποιείται αξιολόγηση των αλγορίθμων που χρησιμοποιούνται, σε πραγματικό ή σε δεύτερο χρόνο, μέσω διάφορων μετρικών αξιολόγησης, οι οποίες επίσης μπορούν να οριστούν, όπως η κάλυψη και η πρωτοπορία. Τέλος, η «Υπηρεσία Παροχής Υπηρεσίας» λειτουργεί ως ένας *διακομιστής μεσολάβησης* (proxy) για τα εισερχόμενα αιτήματα, καθώς για λόγους ασφάλειας αποκόπτεται η απευθείας επικοινωνία των επιμέρους υπηρεσιών.

Ένα άλλο σύστημα παραγωγής συστάσεων σε πραγματικό χρόνο είναι αυτό το οποίο χρησιμοποιείται από την υπηρεσία “Google News” [24]. Τα δεδομένα εισόδου του συστήματος αυτού είναι δεδομένα έμμεσης ανατροφοδότησης δυαδικής μορφής και πιο συγκεκριμένα, αλληλεπιδράσεις (κλικ) των χρηστών με άρθρα, ενώ για τον υπολογισμό των συστάσεων χρησιμοποιείται ένας συνδυασμός αλγορίθμων συνεργατικού φιλτραρίσματος με βάση τη μνήμη και με βάση το μοντέλο. Το σύστημα μπορεί να δεχθεί δύο ειδών αιτημάτων, αιτήματα παροχής συστάσεων και αιτήματα ενημέρωσης των δεδομένων, κάθε φορά που οι χρήστες αλληλεπιδρούν με ένα αντικείμενο (ανοίγουν ένα άρθρο).

Παρόμοια με το σύστημα της “Google News” το «Σύστημα Συστάσεων Ειδήσεων σε Πραγματικό Χρόνο» έχει ως σκοπό την παροχή ποιοτικών, κάθε στιγμή, συστάσεων για άρθρα, ακολουθώντας όμως μία διαφορετική προσέγγιση [25]. Το σύστημα συνδυάζει το μακροπρόθεσμο προφίλ του χρήστη, το περιεχόμενο που περιγράφει την τελευταία επίσκεψη του χρήστη, και το πιο δημοφιλές άρθρο. Το προφίλ του χρήστη κατασκευάζεται με βάση τις συνήθειες περιήγησης του σε βάθος χρόνου (ιστορικό περιήγησης), ενώ το περιεχόμενο περιγράφεται από την ημερομηνία και την τοποθεσία του χρήστη και το τελευταίο άρθρο που διάβασε. Με βάση αυτά τα δεδομένα, το σύστημα προβλέπει κατηγορίες άρθρων, οι οποίες είναι περισσότερο πιθανό να αρέσουν στον χρήστη και στη συνέχεια, από τις τρεις περισσότερο πιθανές, παράγονται συστάσεις με τα δημοφιλέστερα άρθρα των τελευταίων έξι ωρών. Προκειμένου να επιτευχθεί αυτό, τα δεδομένα των χρηστών χωρίζονται σε δύο μέρη, το ένα με δεδομένα χρηστών, οι οποίοι έχουν περισσότερες από 50 επισκέψεις και το άλλο με αυτούς που έχουν λιγότερες από αυτό το όριο. Με αυτόν τον τρόπο, το ιστορικό περιήγησης αποτελεί συνιστώσα στην παραγωγή συστάσεων μόνο των χρηστών της πρώτης ομάδας. Έπειτα, χρησιμοποιώντας τεχνικές μηχανικής μάθησης, προβλέπονται οι πιο ενδιαφέρουσες κατηγορίες άρθρων και από κάθε μία προτείνονται τα καλύτερα νέα για τον χρήστη άρθρα.

Κάθε ένα από αυτά τα συστήματα έχει τα πλεονεκτήματά του αλλά και τις αδυναμίες του. Για παράδειγμα τα δύο τελευταία συστήματα έχουν σχεδιαστεί έτσι ώστε να λειτουργούν, με πολύ αποτελεσματικό τρόπο, μόνο σε εφαρμογές με συγκεκριμένο περιεχόμενο, δηλαδή εφαρμογές παροχής ειδήσεων. Ωστόσο, αυτό πολλές φορές δεν είναι το ιδανικό. Τα συστήματα συστάσεων πρέπει να μπορούν να δουλεύουν με οποιαδήποτε εφαρμογή, ανεξάρτητα από το περιεχόμενό της. Έτσι, με βάση αυτή την παράμετρο, έχουν σχεδιαστεί τα δύο πρώτα συστήματα, που αναφέρθηκαν παραπάνω, “StreamRec” και “ScaR”. Το τελευταίο, όπως αναφέρθηκε νωρίτερα, προσφέρει μεγάλη ευελιξία και πολλές δυνατότητες, σε ότι αφορά τη διαδικασία παραγωγής συστάσεων. Παρόλα αυτά, ο τρόπος με τον οποίο δουλεύει μπορεί να απλουστευτεί σε μεγάλο βαθμό, καθώς ευρεία μερίδα εφαρμογών δεν έχει τόσο υψηλές απαιτήσεις παραμετροποίησης. Αυτή η απλούστευση, από την άλλη, έχει ως αντίκτυπο μία μικρή μείωση στην ποιότητα των συστάσεων, λόγω του ότι θα ενημερώνονται σε σχεδόν πραγματικό χρόνο, κάτι το οποίο είναι αποδεκτό, επίσης από αρκετές εφαρμογές.

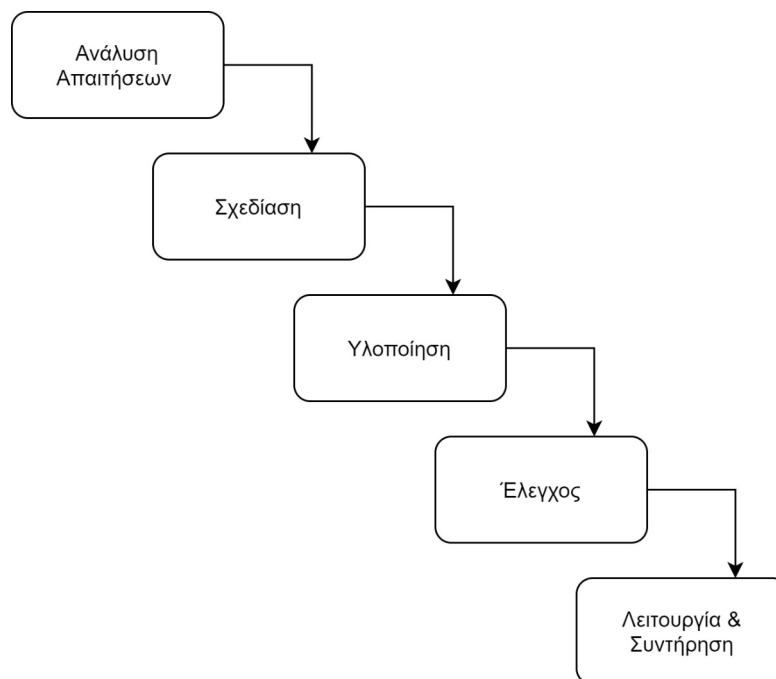
Κεφάλαιο 3

Μεθοδολογία

ΣΤΟ τρέχον κεφάλαιο της εργασίας γίνεται μία αναλυτική περιγραφή της μεθοδολογίας που ακολουθήθηκε για την ανάπτυξη του σχεδόν πραγματικού χρόνου συστήματος συστάσεων. Ειδικότερα, περιγράφεται το μοντέλο ανάπτυξης, το οποίο χρησιμοποιήθηκε, όλες σχεδιαστικές επιλογές που έγιναν μέχρι την τελική μορφή του συστήματος καθώς και τα εργαλεία λογισμικού, τα οποία εν τέλει αξιοποιήθηκαν προκειμένου να επιτευχθεί, σε έναν ικανοποιητικό βαθμό, ο στόχος της εργασίας.

3.1 Μοντέλο Ανάπτυξης

Το μοντέλο ανάπτυξης, το οποίο χρησιμοποιήθηκε για τη δημιουργία του σχεδόν πραγματικού χρόνου συστήματος συστάσεων είναι το *Μοντέλο Καταρράκτη* (Waterfall Model). Το συγκεκριμένο μοντέλο *Κύκλου Ζωής Ανάπτυξης Λογισμικού* (Software Development Life Cycle – SDLC) είναι ένα στατικό μοντέλο και καθορίζει την ανάπτυξη λογισμικού ως μία σειρά διαδοχικών δραστηριοτήτων - φάσεων [26]. Αυτό σημαίνει ότι η έναρξη μίας φάσης προϋποθέτει την ολοκλήρωση της προηγούμενης φάσης και ότι δεν μπορεί να γίνει μετάβαση σε μία προηγούμενη. Κάθε μία έχει το δικό της διαφορετικό στόχο ενώ συνήθως το αποτέλεσμα μίας φάσης χρησιμοποιείται ως είσοδος από την επόμενη.



Εικόνα 3.1: Τα Στάδια Ανάπτυξης του Μοντέλου Καταρράκτη

Οι φάσεις που προβλέπονται από το μοντέλο είναι η ανάλυση των απαιτήσεων, η σχεδίαση, η υλοποίηση, ο έλεγχος και η λειτουργία και η συντήρηση [Εικόνα 3.1]. Αναλυτικότερα, στην πρώτη φάση, συλλέγονται και καταγράφονται όλες οι λειτουργικές και μη-λειτουργικές απαιτήσεις των πελατών για το σύστημα. Έπειτα, κατά το σχεδιασμό του συστήματος, αναλύονται οι απαιτήσεις της προηγούμενης φάσης και ορίζεται ο τρόπος με τον οποίο θα λειτουργεί και η αρχιτεκτονική του συστήματος. Στη φάση της υλοποίησης, με βάση τις σχεδιαστικές παραμέτρους που ελήφθησαν, αναπτύσσεται το σύστημα σε επιμέρους μονάδες (units) (μικρά προγράμματα), η συγχώνευση των οποίων γίνεται στην επόμενη φάση. Επομένως, στη φάση του ελέγχου (συχνά αναφέρεται και ως φάση ενσωμάτωσης και ελέγχου), αρχικά γίνεται έλεγχος των μονάδων που αναπτύχθηκαν και έπειτα πραγματοποιείται έλεγχος του ενιαίου συστήματος, αφού έχει γίνει η συγχώνευση των μονάδων. Τέλος, στο στάδιο της λειτουργίας και του ελέγχου, παρακολουθείται η λειτουργία του συστήματος, αφού έχει παραδοθεί στον πελάτη, εφαρμόζονται διορθώσεις τυχών προβλημάτων και παραδίδονται οι νέες εκδόσεις με τις αλλαγές.

Βασικό πλεονέκτημα του μοντέλου καταρράκτη είναι το γεγονός ότι επιτρέπει την τμηματοποίηση και τον έλεγχο, επιτρέποντας έτσι τον ορισμό χρονοδιαγραμμάτων για την ολοκλήρωση κάθε σταδίου [27]. Επίσης, στα θετικά του συγκεκριμένου μοντέλου είναι η απλότητά του και η ευκολία στη χρήση και διαχείρισή του, καθώς κάθε φάση έχει έναν ακριβή στόχο, και το γεγονός ότι μπορεί να εφαρμοστεί σε μικρά πρότζεκτ, με καλά διατυπωμένες απαιτήσεις, δίνοντας ένα αρκετά καλό αποτέλεσμα. Από την άλλη, το μοντέλο δεν επιτρέπει την πραγματοποίηση αλλαγών. Αυτό έχει ως αποτέλεσμα να καθίσταται ακατάλληλη η εφαρμογή του μοντέλου σε πρότζεκτς, στα οποία υπάρχει μεγάλη πιθανότητα να προκύπτουν νέες απαιτήσεις. Μεταξύ αυτού του μειονεκτήματος βρίσκεται το γεγονός ότι δεν παράγεται κάποια λειτουργική έκδοση του συστήματος, πριν τα τελικά στάδια του μοντέλου, υπάρχει μεγάλο ρίσκο και αβεβαιότητα στο τελικό αποτέλεσμα και δεν είναι κατάλληλο για την ανάπτυξη σύνθετων συστημάτων.

Δεδομένου ότι, για το προς ανάπτυξη σύστημα συστάσεων, έχουν προκαθοριστεί, είναι ξεκάθαρες και δεν πρόκειται να προκύψουν νέες απαιτήσεις για τη λειτουργία του, δεν είναι απαραίτητο ένα επαναληπτικό μοντέλο ανάπτυξης ή η μετάβαση σε προηγούμενα στάδια. Επίσης, δεν απαιτείται η δημιουργία ενδιάμεσων εκδόσεων του εν λόγω συστήματος, μιας και η αξιολόγηση και η βελτίωσή του δεν εμπεριέχεται στα πλαίσια αυτής της εργασίας. Επομένως, για τους παραπάνω λόγους, θεωρείται καλή η χρήση ενός απλού μοντέλου ανάπτυξης, όπως το μοντέλο καταρράκτη.

3.2 Σχεδιασμός του Συστήματος

Όσον αφορά τις απαιτήσεις του συστήματος συστάσεων, το οποίο πρόκειται να αναπτυχθεί, εν μέρη προσδιορίζονται από το γενικότερο στόχο της εργασίας. Έτσι, το επιθυμητό σύστημα θα πρέπει να παράγει κάθε χρονική στιγμή ακριβείς και ενδιαφέρουσες για τον χρήστη συστάσεις, χωρίς η ποιότητα αυτών να επηρεάζεται από τις αλλαγές στο περιεχόμενο της εφαρμογής. Επίσης, το σύστημα θα πρέπει να έχει τη μορφή ενός εργαλείου, το οποίο να μπορεί να ενσωματωθεί σε οποιαδήποτε εφαρμογή, χωρίς επιπλέον τροποποιήσεις του, ανεξάρτητα από τον τύπο των αντικειμένων της. Προκειμένου να επιτευχθούν οι στόχοι αυτοί και να λειτουργεί το σύστημα με αυτό τον τρόπο, θα πρέπει να γίνουν οι κατάλληλες σχεδιαστικές επιλογές σε ότι έχει να κάνει με την τεχνική παραγωγής συστάσεων και τον αλγόριθμο που θα χρησιμοποιηθεί, το είδος των δεδομένων, τα οποία θα δέχεται ως είσοδο και τον τρόπο μεταφοράς των δεδομένων ανάμεσα στα διάφορα μέρη του συστήματος.

Αρχικά, με σκοπό την παροχή ποιοτικών συστάσεων με βάση τις αλλαγές στο περιεχόμενο της εφαρμογής, θα πρέπει το σύστημα να ενημερώνεται διαρκώς και άρα θα πρέπει τα δεδομένα εισόδου να έχουν τη μορφή δεδομένων συνεχούς ροής. Προκειμένου να επιτευχθεί αυτό, απαιτείται ένας ευέλικτος και αποτελεσματικός τρόπος, με τον οποίο θα μεταφέρονται τα δεδομένα από την εφαρμογή και θα τροφοδοτούνται στο σύστημα. Αυτό, μπορεί να πραγματοποιηθεί χρησιμοποιώντας το μοτίβο ανταλλαγής μηνυμάτων Publish / Subscribe, το οποίο περιγράφηκε στο προηγούμενο κεφάλαιο, όπου η εφαρμογή θα προωθεί τα δεδομένα σε ένα θέμα

και την ίδια στιγμή, μόνο τα κατάλληλα μέρη του συστήματος θα λαμβάνουν αυτά τα δεδομένα, όντας εγγεγραμμένα στο θέμα αυτό.

Αναφορικά με την τεχνική, την οποία θα χρησιμοποιεί το σύστημα ώστε να παράγει συστάσεις, το συνεργατικό φιλτράρισμα είναι μία επιλογή, που μπορεί να υποστηρίξει το σκοπό του συστήματος. Άλλωστε, η χρήση μιας τεχνικής που βασίζεται στο περιεχόμενο της εφαρμογής, θα ήταν εμπόδιο για την εκπλήρωση της λειτουργικής απαίτησης, η οποία θέλει το σύστημα να λειτουργεί σε οποιαδήποτε εφαρμογή. Παράλληλα, το είδος των δεδομένων εισόδου, με το οποίο θα τροφοδοτείται το σύστημα, είναι δεδομένα έμμεσης ανατροφοδότησης, μιας και έχουν μεγαλύτερη διαθεσιμότητα σε σχέση με τα δεδομένα ρητής ανατροφοδότησης, ενώ δίνεται μεγαλύτερη ευελιξία στις εφαρμογές, καθώς μπορούν να τα ερμηνεύουν όπως εκείνες επιθυμούν. Αυτό σημαίνει πως η κάθε εφαρμογή μπορεί να ορίζει την αλληλεπίδραση ενός χρήστη με ένα αντικείμενο με διαφορετικό τρόπο (π.χ. χρόνος ανάγνωσης, συχνότητα αγοράς, κ.ο.κ.).

Ωστόσο, οι αλγόριθμοι συνεργατικού φιλτραρίσματος, δεν είναι σχεδιασμένοι ώστε να δουλεύουν με δεδομένα έμμεσης ανατροφοδότησης και άρα θα πρέπει να γίνονται κάποιοι μετασχηματισμοί. Τα δεδομένα έμμεσης ανατροφοδότησης, όπως αναφέρεται και στο προηγούμενο κεφάλαιο, δεν εμπεριέχουν αρνητική ανατροφοδότηση. Το γεγονός ότι ένας χρήστης δεν έχει αλληλεπιδράσει καμία φορά με ένα συγκεκριμένο αντικείμενο, μπορεί να υποδηλώνει ότι δεν του αρέσει ή ότι δε γνωρίζει ακόμα για την ύπαρξή του. Επομένως, τα δεδομένα έμμεσης ανατροφοδότησης αντικατοπτρίζουν την εμπιστοσύνη και όχι την προτίμηση. Όσες, περισσότερες είναι οι αλληλεπιδράσεις r_{ui} ενός χρήστη u με ένα αντικείμενο i , τόσο πιο πιθανό είναι να του αρέσει. Η τιμή της εμπιστοσύνης c_{ui} ορίζεται ως εξής [7]:

$$c_{ui} = 1 + ar_{ui}$$

Με αυτό τον τρόπο, υπάρχει μία ελάχιστη τιμή εμπιστοσύνης για την προτίμηση κάθε ζεύγους χρήστη - αντικειμένου, η οποία αυξάνεται καθώς προκύπτουν περισσότερες αλληλεπιδράσεις – στοιχεία θετικής προτίμησης. Ο ρυθμός αύξησης ελέγχεται από τον παράγοντα a .

Επίσης, πρέπει να αναφερθεί ότι υιοθετώντας την τεχνική του συνεργατικού φιλτραρίσματος, ως τον τρόπο με τον οποίο το σύστημα θα παράγει συστάσεις, αυτομάτως κληρονομούνται και όλα προβλήματα που προκύπτουν με τη χρήση αυτής της τεχνικής. Αναλυτικότερα, το βασικό πρόβλημα της είναι το γεγονός, ότι δεν μπορούν να γίνουν ακριβείς συστάσεις σε έναν νέο χρήστη ή για ένα νέο αντικείμενο, δεδομένου ότι δεν έχουν ληφθεί αρκετά δεδομένα. Το πρόβλημα αυτό αναλύθηκε στο προηγούμενο κεφάλαιο ως το πρόβλημα κρύας εκκίνησης. Παρόλα αυτά, δεδομένου ότι η αντιμετώπιση του προβλήματος δεν είναι ο στόχος του συστήματος στην παρούσα φάση, δεν λαμβάνονται κατασταλτικά μέτρα. Πρακτικά, οι νέοι χρήστες λαμβάνουν στην αρχή συστάσεις για τυχαία αντικείμενα και τα νέα αντικείμενα προτείνονται στους χρήστες με τυχαίο τρόπο.

Η επόμενη σχεδιαστική επιλογή, σχετίζεται με το είδος του συνεργατικού φιλτραρίσματος, το οποίο θα γίνεται από το σύστημα. Το συνεργατικό φιλτράρισμα με βάση τη μνήμη, αποδεικνύεται μια καλή λύση, όμως όσο αυξάνεται ο όγκος των δεδομένων, τόσο πιο απαιτητικοί γίνονται οι υπολογισμοί. Κάτι τέτοιο μπορεί να επηρεάσει την ποιότητα των συστάσεων, καθώς το σύστημα αδυνατεί να ενημερώνεται σε πραγματικό χρόνο λόγω των χρονοβόρων υπολογισμών. Αντιθέτως, το συνεργατικό φιλτράρισμα με βάση το μοντέλο μπορεί να δώσει μία λύση σε αυτό το πρόβλημα. Χρησιμοποιώντας μία μέθοδο μείωσης των διαστάσεων του πίνακα χρηστών – αντικειμένων (παραγοντοποίηση πίνακα), οι υπολογισμοί των συστάσεων (εκπαίδευση μοντέλου), μέσα από δεδομένα μεγάλου όγκου, επιταχύνονται. Για αυτό τον σκοπό, ο αλγόριθμος Alternating Least Squares θα χρησιμοποιείται από το σύστημα, μιας και η χρήση του είναι ευρέως διαδεδομένη και παράγει ικανοποιητικά αποτελέσματα.

Παρόλα αυτά, μολονότι χρησιμοποιείται η μέθοδος μείωσης των διαστάσεων, ο διαρκώς αυξανόμενος όγκος των δεδομένων αλληλεπιδράσεων, που μπορεί να προκύψει σε βάθος χρόνου σε μία εφαρμογή, μπορεί και πάλι να κάνει τους υπολογισμούς αρκετά απαιτητικούς. Το πρόβλημα αυτό ενισχύεται στην περίπτωση του τρέχοντος συστήματος, το οποίο εκπαιδεύει από την αρχή το μοντέλο κάθε φορά που λαμβάνει νέα

δεδομένα, προκειμένου να παράγει ακριβείς συστάσεις. Μία λύση, μπορεί να δοθεί λαμβάνοντας υπόψιν στον υπολογισμό κάθε φορά, μόνο ένα ορισμένο πλήθος των πιο πρόσφατων αλληλεπιδράσεων. Επομένως, πραγματοποιώντας ένα είδος επεξεργασίας τύπου “mini-batch processing” στα συνεχούς ροής δεδομένα και εφαρμόζοντας μία μέθοδο ενός κυλιόμενου παραθύρου, μπορεί να οριοθετηθεί ο χρόνος των υπολογισμών και άρα η ενημέρωση των συστάσεων του συστήματος να γίνεται σε σχεδόν πραγματικό χρόνο. Το μήκος του παραθύρου, δηλαδή ο αριθμός των τελευταίων αλληλεπιδράσεων και ο αριθμός των νέων αλληλεπιδράσεων, οι οποίες πρέπει να ληφθούν ώστε να γίνει εκ νέου εκπαίδευση του μοντέλου με τα δεδομένα του επόμενου παραθύρου, είναι παράμετροι που δίνεται η δυνατότητα στην κάθε εφαρμογή να ρυθμίσει, με βάση τις ανάγκες της.

Τέλος, έχοντας περιγράψει τον τρόπο με τον οποίο τα δεδομένα εισόδου θα μεταφέρονται από την εφαρμογή προς το σύστημα, το είδος των δεδομένων καθώς και τη διαδικασία της επαναληπτικής εκπαίδευσης του μοντέλου συστάσεων, απομένει ο σχεδιασμός του τμήματος παροχής των συστάσεων στην εφαρμογή. Στο σημείο αυτό πρέπει να αναφερθεί ότι, για την καλύτερη λειτουργία και συντήρηση του, το σύστημα χωρίζεται σε δύο μέρη. Ένα μέρος υπεύθυνο για τη λήψη των δεδομένων εισόδου και παράλληλα τη διαρκή παραγωγή του μοντέλου συστάσεων και ένα δεύτερο μέρος, το οποίο λειτουργεί ως ένας «εξυπηρετητής» συστάσεων. Έτσι, το δεύτερο μέρος αναλαμβάνει την διανομή των συστάσεων προς την εφαρμογή, ύστερα από σχετικό αίτημά της. Πιο συγκεκριμένα, η εφαρμογή στέλνει ένα αίτημα για παροχή συστάσεων στον εξυπηρετητή, ορίζοντας τον χρήστη, για τον οποίο προορίζονται οι συστάσεις, καθώς και τον αριθμό των συστάσεων (αντικειμένων) που πρέπει να επιστραφούν. Έχοντας λάβει το αίτημα, ο εξυπηρετητής χρησιμοποιώντας το πιο πρόσφατα παραγόμενο μοντέλο, υπολογίζει τα αντικείμενα, τα οποία είναι πιθανότερο να αρέσουν στο συγκεκριμένο χρήστη και τα στέλνει στην εφαρμογή.

3.2.1 Αλγόριθμος Alternating Least Squares

Η παραγοντοποίηση πίνακα, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο είναι μία δημοφιλής μέθοδος, η οποία για ένα σχετικά μικρό αριθμό k , συνοψίζει κάθε χρήστη σε ένα πίνακα k διαστάσεων x_u και κάθε αντικείμενο i σε έναν πίνακα y_i . Οι πίνακες αυτοί αποκαλούνται κρυφοί παράγοντες. Για να προβλεφτεί μία βαθμολογία ενός χρήστη για ένα αντικείμενο αρκεί η πρόβλεψη του $r_{ui} = x_u^T y_i$. Βάζοντας σε έναν πίνακα X με διαστάσεις $k \times n$, όπου n το πλήθος των χρηστών, τους κρυφούς παράγοντες όλων των χρηστών και σε έναν πίνακα Y διαστάσεων $m \times k$, όπου m το πλήθος των αντικειμένων, του κρυφούς παράγοντες των αντικειμένων, ο στόχος είναι η συμπλήρωση των κενών του πίνακα $R = X^T Y$. Αυτό μπορεί να εκφραστεί ως ένα πρόβλημα βελτιστοποίησης, στο οποίο ο στόχος είναι η εύρεση των X και Y , ελαχιστοποιώντας το σφάλμα μεταξύ των προβλεπόμενων και των πραγματικών (γνωστών) βαθμολογιών ως εξής [28]:

$$\min_{X, Y} \sum_{\text{γνωστή } r_{ui}} (r_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

όπου λ παράγοντας κανονικοποίησης του μοντέλου, έτσι ώστε να μη γίνει υπερ-εξειδίκευση με τα δεδομένα εκπαίδευσης.

Το πρόβλημα μπορεί να λυθεί κρατώντας σταθερό τον πίνακα Y και βελτιστοποιώντας τον πίνακα X , έπειτα κρατώντας σταθερό τον X και βελτιστοποιώντας τον Y και επαναλαμβάνοντας αυτή την διαδικασία μέχρι τη μείωση του σφάλματος σε ένα ικανοποιητικό επίπεδο [10]. Η μέθοδος αυτή είναι ο αλγόριθμος Alternating Least Squares, ο οποίος θα χρησιμοποιείται στο τρέχον σύστημα συστάσεων.

Ωστόσο, στην περίπτωση που χρησιμοποιούνται δεδομένα έμμεσης ανατροφοδότησης, θα πρέπει να γίνουν κάποιες τροποποιήσεις στο πρόβλημα της βελτιστοποίησης. Πιο συγκεκριμένα, είναι απαραίτητο να συμπεριλαμβάνονται στον υπολογισμό και τα διάφορα επίπεδα εμπιστοσύνης που υπάρχουν, ενώ θα πρέπει επίσης να συμπεριλαμβάνονται όλα τα ζεύγη χρηστών – αντικειμένου και όχι μόνο τα ζεύγη για τα οποία υπάρχουν τιμές [7]. Επομένως, η ελαχιστοποίηση του σφάλματος γίνεται ως εξής:

$$\min_{X,Y} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

3.3 Εργαλεία που Χρησιμοποιήθηκαν

Για την επίτευξη των στόχων της εργασίας και τη δημιουργία του σχεδόν πραγματικού χρόνου συστήματος συστάσεων, έγινε η χρήση διάφορων εργαλείων όπως γλώσσες προγραμματισμού, βιβλιοθήκες κώδικα, περιβάλλοντα ανάπτυξης και άλλου σχετικού λογισμικού. Παρακάτω, αναλύεται ο τρόπος με τον οποίο αξιοποιήθηκε το κάθε εργαλείο και δίνεται μία περιγραφή των βασικών χαρακτηριστικών και των λειτουργιών τους.

3.3.1 Python 3.8

Η ανάπτυξη των προγραμμάτων του συστήματος έγινε χρησιμοποιώντας τη γλώσσα προγραμματισμού Python και συγκεκριμένα την έκδοση 3.8. Αναλυτικότερα, με αυτή τη γλώσσα αναπτύχθηκε το πρόγραμμα, το οποίο λαμβάνει τα δεδομένα αλληλεπιδράσεων των χρηστών, πραγματοποιεί τους κατάλληλους μετασχηματισμούς, ώστε αυτά να έρθουν σε μία επιθυμητή μορφή και τέλος εκπαιδεύει και εξάγει το μοντέλο συστάσεων. Επίσης, στη γλώσσα αυτή έγινε η συγγραφή του κώδικα του προγράμματος, το οποίο υπολογίζει, με τη χρήση του μοντέλου τα υποψήφια προτεινόμενα αντικείμενα για κάθε χρήστη και τα προωθεί στην εφαρμογή. Τέλος, αναπτύχθηκαν διάφορα βοηθητικά προγράμματα προκειμένου να προσομοιωθούν σε ένα βαθμό οι συνθήκες και τα δεδομένα μίας πραγματικής εφαρμογής. Η λειτουργία των προγραμμάτων αυτών περιγράφεται αναλυτικά στο επόμενο κεφάλαιο.

Η επιλογή αυτής της γλώσσας έγινε διότι αποτελεί μία ευρέως διαδεδομένη επιλογή για την ανάπτυξη εφαρμογών που στηρίζονται σε τεχνολογίες μηχανικής μάθησης, ανήκουν δηλαδή στην περιοχή της *Επιστήμης των Δεδομένων* (Data Science) αλλά και για το πλήθος των διαθέσιμων βιβλιοθηκών – πακέτων ανοιχτού κώδικα.



Εικόνα 3.2: Το Λογότυπο της Python [29]

Γενικότερα, η Python είναι μία *διερμηνευόμενη* (interpreted), *αντικειμενοστραφής* (object-oriented), υψηλού επιπέδου, γενικής χρήσης γλώσσα προγραμματισμού και μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού παγκοσμίως. Υποστηρίζει διάφορα προγραμματιστικά παραδείγματα, πέρα από τον αντικειμενοστραφή προγραμματισμό, όπως το *συναρτησιακό* (functional) και *διαδικαστικό* (procedural) προγραμματισμό [29]. Η Python περιλαμβάνει έναν «καθαρό» τρόπο σύνταξης ενώ ταυτόχρονα παρέχει πολλαπλές δυνατότητες. Επίσης, περιέχει διεπαφές για πολλές κλήσεις συστήματος, μεγάλη ποικιλία σε βιβλιοθήκες καθώς επίσης μπορεί να χρησιμοποιηθεί ως μία γλώσσα επέκτασης σε εφαρμογές που χρειάζονται μία *προγραμματιστική διεπαφή* (programming interface). Η Python μπορεί να εκτελείται σε πολλές παραλλαγές του Unix, όπως τις διανομές Linux και το macOS, αλλά και σε συστήματα με λειτουργικό σύστημα Windows.

Βασικό χαρακτηριστικό στο σχεδιασμό γλώσσας είναι η έμφαση που δίνεται στην ευαναγνωσιμότητα. Η δομή της γλώσσας και η αντικειμενοστραφής της προσέγγιση προτρέπει τους προγραμματιστές στο να γράφουν εύκολα κατανοητό και ευανάγνωστο κώδικα, τόσο σε μικρά όσο και σε μεγάλα πρότζεκτς. Επίσης, στα θετικά χαρακτηριστικά της γλώσσας συγκαταλέγεται το γεγονός ότι είναι αρκετά εύχρηστη, καθιστώντας έτσι απλή τη δημιουργία προγραμμάτων. Αυτό σημαίνει ότι η γλώσσα είναι ιδανική για την

ανάπτυξη πρωτότυπων λειτουργιών προγραμματισμού, χωρίς να διακυβεύεται η δυνατότητα εύκολης συντήρησής τους. Επιπλέον θετικά στοιχεία της Python, είναι η διαδραστική της λειτουργία, που επιτρέπει τη δοκιμή σύντομων αποσπασμάτων κώδικα, η δυνατότητα επέκτασής της, με την προσθήκη νέων ενοτήτων, οι οποίες μπορούν να υλοποιηθούν με μία μεταγλωττισμένη γλώσσα όπως τη C ή τη C++, και το γεγονός ότι είναι ελεύθερη, δηλαδή διαθέσιμη με άδεια ανοικτού κώδικα (open source).

Η Python είναι μία γλώσσα προγραμματισμού γενικού σκοπού, η οποία μπορεί να εφαρμοστεί σε διάφορες κατηγορίες προβλημάτων. Η γλώσσα παρέχει μία μεγάλη βασική βιβλιοθήκη που υποστηρίζει πολλές κοινές εργασίες προγραμματισμού όπως σύνδεση σε διακομιστές διαδικτύου, αναζήτηση κειμένου με κανονικές εκφράσεις, ανάγνωση, τροποποίηση και υπολογισμό διαφορών μεταξύ αρχείων. Ωστόσο, είναι διαθέσιμη μία μεγάλη ποικιλία επεκτάσεων της, με τη μορφή πακέτων, τα οποία επεκτείνουν την εφαρμογή της γλώσσας.

3.3.2 Βιβλιοθήκες της Python

Όπως αναφέρθηκε νωρίτερα, υπάρχουν διάφορες βιβλιοθήκες της Python διαθέσιμες, με την κάθε μία να εξυπηρετεί ένα διαφορετικό σκοπό. Οι βιβλιοθήκες ή “modules” όπως αποκαλούνται στην Python, εμπεριέχονται σε πακέτα (packages). Για τη διαχείριση αυτών των πακέτων, δηλαδή για την εύρεση, κατηγοριοποίηση, εγκατάσταση, ενημέρωσή τους, υπάρχουν διάφοροι διαχειριστές πακέτων (package managers), με τον πιο δημοφιλή να είναι το Πρόγραμμα Εγκατάστασης Πακέτων Python (Python Package Installer - PIP).

Για τις ανάγκες της υλοποίησης του σχεδόν πραγματικού χρόνου συστήματος συστάσεων, χρησιμοποιήθηκαν διάφορα πακέτα Python. Ανάμεσα σε αυτά είναι το πακέτο “**kafka-python**”, το οποίο αποτελεί έναν “client”, για τη καταναεμημένη πλατφόρμα μετάδοσης συνεχούς ροής γεγονότων Apache Kafka (περιγράφεται αναλυτικότερα στην επόμενη ενότητα). Με άλλα λόγια, αποτελεί μία διεπαφή για την πλατφόρμα αυτή, επιτρέποντας την χρήση της μέσα από κώδικα Python (π.χ. εγγραφή σε θέματα, προώθηση και λήψη μηνυμάτων από ένα θέμα). Η χρήση της στο τρέχον σύστημα κρίνεται απαραίτητη, καθώς γίνεται χρήση του Apache Kafka για την μεταγωγή των συνεχούς ροής δεδομένων μεταξύ των διάφορων μερών του συστήματος.

Για την υλοποίηση του τμήματος του συστήματος, το οποίο είναι υπεύθυνο για την παραγωγή των συστάσεων, χρησιμοποιήθηκε το πακέτο “**Implicit**” [30]. Το συγκεκριμένο πακέτο παρέχει υλοποιήσεις για διάφορους αλγόριθμους παραγωγής συστάσεων με δεδομένα έμμεσης ανατροφοδότησης, ανάμεσα στους οποίους βρίσκεται και ο Alternating Least Squares. Παρέχονται συναρτήσεις για πολυ-νηματική (multi-threading) εκπαίδευση του μοντέλου, για χρήση αυτού για την παραγωγή συστάσεων, ενώ δίνεται η δυνατότητα παραμετροποίησης των αλγορίθμων.

Άλλο ένα πακέτο της Python, το οποίο χρησιμοποιήθηκε για την υλοποίηση του συστήματος είναι το “**Flask**”. Το συγκεκριμένο πακέτο αποτελεί ένα “web framework”, το οποίο επιτρέπει την ανάπτυξη εφαρμογών δικτύου (web applications). Στην προκειμένη περίπτωση, χρησιμοποιήθηκε για τη δημιουργία ενός εξυπηρετητή συστάσεων, δηλαδή του μέρους του συστήματος που είναι υπεύθυνο για την παροχή συστάσεων για την εφαρμογή.

Ανάμεσα σε αυτά τα τρία βασικά πακέτα που προαναφέρθηκαν, έχουν χρησιμοποιηθεί και άλλα με μικρότερο αλλά εξίσου καθοριστικό ρόλο. Ενδεικτικά, γίνεται χρήση της βιβλιοθήκης “**pandas**”, η οποία χρησιμοποιείται για τον χειρισμό και την ανάλυση δεδομένων. Πιο συγκεκριμένα, παρέχει ειδικές δομές δεδομένων και λειτουργίες για τον χειρισμό αριθμητικών πινάκων και χρονοσειρών. Στην τρέχουσα υλοποίηση η βιβλιοθήκη είναι απαραίτητη για την μετατροπή των δεδομένων εισόδου σε κατάλληλη μορφή, έτσι ώστε να τροφοδοτούνται στο μοντέλο για εκπαίδευση. Επιπροσθέτως, προκειμένου τα δεδομένα να αποκτήσουν την κατάλληλη μορφή, ενός δηλαδή αραιού πίνακα χρηστών – αντικειμένων, γίνεται χρήση του εργαλείου “**scipy.sparse**” της βιβλιοθήκης “**SciPy**”. Επίσης, αξιοποιείται η βιβλιοθήκη “**threading**”, έτσι ώστε το πρόγραμμα λήψης των δεδομένων και εκπαίδευσης του μοντέλου να πραγματοποιεί παράλληλα αυτές τις δύο λειτουργίες, χωρίς να χάνονται τυχόν νέα δεδομένα, τα οποία προκύπτουν κατά τη φάση εκπαίδευσης του μοντέλου. Τέλος, με τη βοήθεια του module “**pickle**”, γίνεται η αποθήκευση του μοντέλου

σε δυαδική μορφή (serialization) και η ανάκτησή του (deserialization) όταν αυτή είναι απαραίτητη για τον υπολογισμό των συστάσεων.

3.3.3 Apache Kafka

Το Apache Kafka είναι ένα σύστημα ανταλλαγής μηνυμάτων, το οποίο ακολουθεί το μοτίβο ανταλλαγής μηνυμάτων Publish / Subscribe, σχεδιασμένο για να δώσει μία λύση στις εφαρμογές που έρχονται αντιμέτωπες με συνεχούς ροής όγκους πληροφοριών και επιθυμούν τη γρήγορη δρομολόγησή τους σε πολλαπλούς καταναλωτές [31]. Το σύστημα αναφέρεται και ως μία ανοιχτού κώδικα *κατανεμημένη πλατφόρμα μετάδοσης συνεχούς ροής γεγονότων* (distributed event streaming platform), ενώ χρησιμοποιείται για τη δημιουργία υψηλής απόδοσης “data pipelines”, την ανάλυση συνεχούς ροής δεδομένων, την συγχώνευση των δεδομένων [32]. Τα δεδομένα στο Kafka αποθηκεύονται, προκειμένου να μπορούν να διαβαστούν εν ευθέτω χρόνω, ενώ μπορούν επίσης να κατανεμηθούν μέσα στο σύστημα, έτσι ώστε να αυξάνονται οι επιδόσεις του αλλά και να παρέχεται επιπλέον προστασία στην περίπτωση που προκύψουν πιθανά σφάλματα [16].

Η βασική μονάδα δεδομένων στο Apache Kafka είναι το μήνυμα. Για το σύστημα, πρόκειται για μια σειρά από bytes, χωρίς να δίνεται σημασία στη μορφή ή στην ερμηνεία τους. Προαιρετικά, ένα μήνυμα μπορεί να περιλαμβάνει ένα επιπλέον μέρος πληροφορίας (μεταδεδομένο), το οποίο αναφέρεται ως κλειδί. Τα δεδομένα, γράφονται στο σύστημα σε ομάδες (batches), οι οποίες αποτελούν μία συλλογή με όλα τα μηνύματα που έχουν παραχθεί σε ένα (ίδιο) θέμα.

Τα μηνύματα στο Kafka κατηγοριοποιούνται σε θέματα και αυτά με τη σειρά τους χωρίζονται σε έναν αριθμό από *μέρη* (partitions). Αυτή η τμηματοποίηση είναι ένας τρόπος με τον οποίο το Kafka παρέχει πλεονασμό (redundancy) και κλιμακωσιμότητα (scalability), καθώς κάθε μέρος μπορεί να βρίσκεται σε διαφορετικό εξυπηρετητή.

Οι βασικοί πελάτες – χρήστες του Apache Kafka, διακρίνονται σε δύο κατηγορίες, στους παραγωγούς πληροφορίας και στους καταναλωτές της. Οι παραγωγοί παράγουν και προωθούν νέα μηνύματα σε ένα συγκεκριμένο θέμα, ενώ οι καταναλωτές μπορούν να είναι εγγεγραμμένοι σε αυτό ή και σε άλλα θέματα και να λαμβάνουν τα αυτά τα μηνύματα με τη σειρά που παράχθηκαν. Προκειμένου το σύστημα να γνωρίζει ποιο ήταν το τελευταίο μήνυμα που έχει λάβει κάθε καταναλωτής, προστίθεται ένα ακόμα μεταδεδομένο, το “offset”, το οποίο είναι ένας συνεχώς αυξανόμενος ακεραίος αριθμός.



Εικόνα 3.3: Το Λογότυπο του Apache Kafka [32]

Ένας εξυπηρετητής Kafka ονομάζεται “broker” και ο ρόλος τους είναι να λαμβάνει μηνύματα από τους παραγωγούς, να αναθέτει offsets σε αυτά και να αποθηκεύει τα μηνύματα στο δίσκο. Αντίστοιχα, προωθεί τα αποθηκευμένα δεδομένα στους καταναλωτές όταν εκείνοι τα ζητήσουν. Παρόλο που ένας εξυπηρετητής Kafka μπορεί να λειτουργεί μόνος τους χωρίς προβλήματα, όσο ο όγκος των δεδομένων βρίσκεται εντός κάποιων ορίων, οι Kafka brokers είναι σχεδιασμένοι για να λειτουργούν ως *ομάδα* (cluster), έτσι ώστε να αυξάνεται η απόδοση και οι δυνατότητές του Kafka.

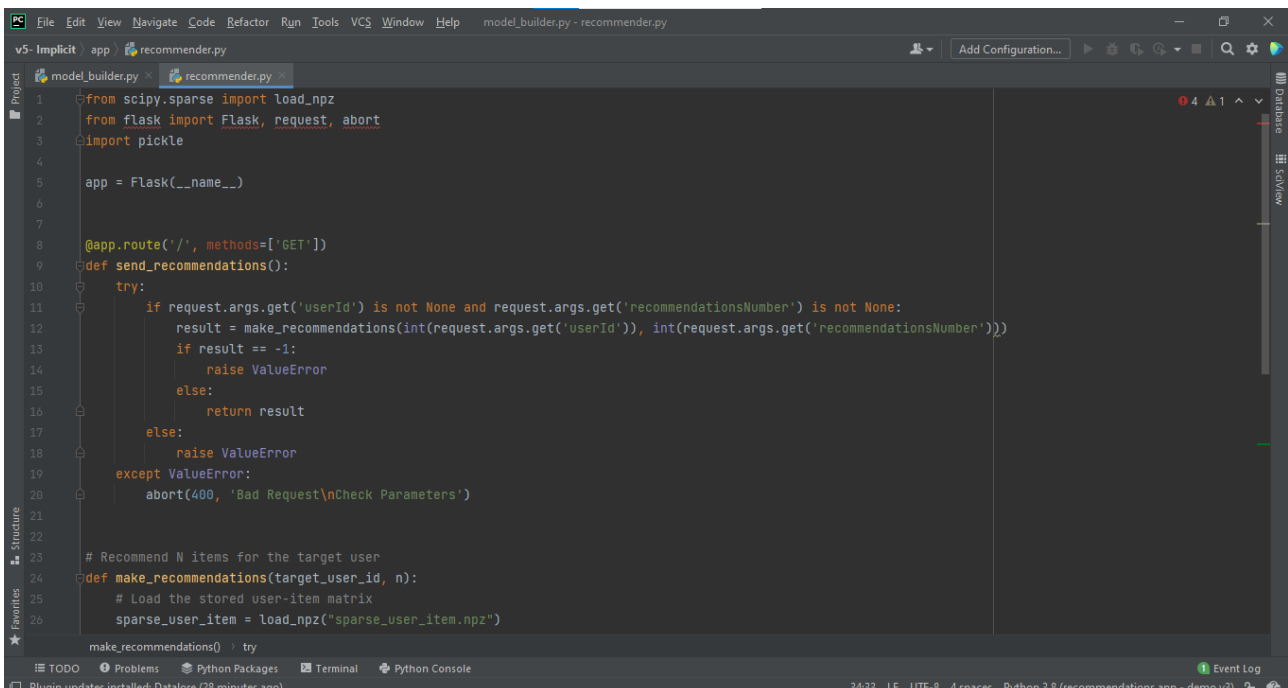
Ωστόσο, για τη λειτουργία του Apache Kafka είναι απαραίτητη μία άλλη υπηρεσία, η οποία ονομάζεται Apache ZooKeeper. Η ZooKeeper είναι μία κεντρική υπηρεσία υπεύθυνη για τη διατήρηση των πληροφοριών των παραμετροποιήσεων, την ονοματοδοσία, και την παροχή καταναμεμημένου συγχρονισμού και υπηρεσιών σε ομάδες. Με λίγα λόγια, η υπηρεσία είναι μία υψηλής απόδοσης υπηρεσία συντονισμού καταναμεμημένων εφαρμογών. Το Kafka χρησιμοποιεί την υπηρεσία για το συντονισμό του Kafka broker και των καταναλωτών.

Όσον αφορά το ρόλο του Apache Kafka στο τρέχον σύστημα συστάσεων, αυτός είναι η μεταφορά των συνεχούς ροής δεδομένων από και προς τα κατάλληλα κάθε φορά μέρη του συστήματος. Πιο συγκεκριμένα, η εφαρμογή (παραγωγός), έχοντας ορίσει την ερμηνεία της αλληλεπίδρασης ενός χρήστη με ένα αντικείμενο, προωθεί τις αλληλεπιδράσεις των χρηστών σε ένα θέμα, τη στιγμή που συμβαίνουν. Έπειτα, το τμήμα του συστήματος, το οποίο λαμβάνει και εκπαιδεύει το μοντέλο συστάσεων (καταναλωτής), όντας εγγεγραμμένο στο συγκεκριμένο θέμα, λαμβάνει άμεσα τα δεδομένα.

3.3.4 Περιβάλλον Ανάπτυξης PyCharm

Για την συγγραφή του κώδικα Python των επιμέρους προγραμμάτων του σχεδόν πραγματικού χρόνου συστήματος συστάσεων, χρησιμοποιήθηκε το *Ολοκληρωμένο Περιβάλλον Ανάπτυξης* (Integrated Development Environment – IDE) PyCharm. Μέσω του επεξεργαστή πηγαιού κώδικα, ο οποίος διατίθεται, έγινε η συγγραφή του κώδικα των προγραμμάτων, ενώ παράλληλα πραγματοποιήθηκε έλεγχος του κώδικα για συντακτικά και λογικά λάθη. Στην εύρεση των λογικών λαθών, καθοριστικό ρόλο είχε το εργαλείο αποσφαλμάτωσης (debugging) που επίσης είναι ενσωματωμένο στο λογισμικό.

Το PyCharm είναι λογισμικό αναπτυγμένο από την εταιρεία JetBrains (ή αλλιώς IntelliJ) και πέρα από τις βασικές λειτουργίες ενός τέτοιου είδους συστήματος, παρέχει ανάλυση του κώδικα, αποσφαλμάτωση με γραφικό περιβάλλον, ενσωμάτωση *συστημάτων ελέγχου εκδόσεων* (version control systems), υποστηρίζει την ανάπτυξη web εφαρμογών με τη χρήση διάφορων web frameworks και εφαρμογών που ανήκουν στο πεδίο της επιστήμης των δεδομένων.



Εικόνα 3.4: Το Περιβάλλον του Λογισμικού PyCharm

Το λογισμικό διανέμεται ελεύθερα (Apache License) και διατίθεται για λειτουργικά συστήματα Windows, macOS και διανομές του Linux.

3.3.5 VMware Player

Τέλος, για την αποτελεσματικότερη και απρόσκοπτη λειτουργία των εργαλείων, η ανάπτυξη του συστήματος συστάσεων, επιλέχθηκε να γίνει σε μία διανομή του λειτουργικού συστήματος Linux. Πιο συγκεκριμένα, έγινε χρήση της διανομής Ubuntu 18.04, λόγω της υψηλής σταθερότητας και αξιοπιστίας που προσφέρει, σε συνδυασμό με το εύχρηστο και ευέλικτο γραφικό του περιβάλλον. Έτσι, στο σύστημα αυτό έγινε εγκατάσταση του προαναφερόμενου περιβάλλοντος ανάπτυξης, του Apache Kafka, της κατάλληλης έκδοσης της γλώσσας Python καθώς και των απαραίτητων βιβλιοθηκών της, οι οποίες περιγράφηκαννωρίτερα.

Δεδομένου ότι κατά την περίοδο ανάπτυξης του συστήματος το βασικό λειτουργικό σύστημα δεν ήταν η συγκεκριμένη διανομή Linux, για πρακτικούς λόγους χρησιμοποιήθηκε το εργαλείο *εικονικοποίησης* (virtualization) VMware Player. Το εργαλείο, το οποίο αναπτύχθηκε από την εταιρεία VMware και διανέμεται ελεύθερα, δίνει την δυνατότητα προσομοίωσης *υλικού* (hardware). Σε μία εικονική μηχανή, όπως ονομάζεται ένα σύνολο εικονικοποιημένου υλικού, μπορεί να γίνει εγκατάσταση ενός οποιουδήποτε λειτουργικού συστήματος και να εκτελείται παράλληλα με το κύριο λειτουργικό σύστημα. Τέλος, πρέπει να αναφερθεί, ότι εργαλείο δίνει τη δυνατότητα παραμετροποίησης της εικονικής μηχανής και μετά την αρχική της δημιουργία (π.χ. αύξηση του μεγέθους της εικονικής κύριας μνήμης, αλλαγή του αριθμού των πυρήνων του εικονικού επεξεργαστή, κ.ο.κ). Αυτό, φάνηκε ιδιαίτερος χρήσιμο στις περιπτώσεις όπου προέκυψε η ανάγκη για περισσότερους υπολογιστικούς πόρους κατά τη διάρκεια ανάπτυξης και δοκιμής του συστήματος συστάσεων.

3.4 Απορριφθείσες Σχεδιαστικές Επιλογές και Εργαλεία

Οι σχεδιαστικές επιλογές και τα εργαλεία, τα οποία χρησιμοποιήθηκαν και περιγράφηκαν στις προηγούμενες ενότητες, προέκυψαν κατόπιν διάφορων αλλαγών κατά τη διάρκεια της φάσης του σχεδιασμού του συστήματος. Πιο συγκεκριμένα, η αρχική ιδέα για την επεξεργασία των συνεχούς ροής δεδομένων ήταν η χρήση μιας μηχανής επεξεργασίας συνεχούς ροής δεδομένων, του Apache Spark.

Το Apache Spark είναι μία ανοικτού κώδικα μηχανή επεξεργασίας δεδομένων μεγάλου όγκου, η οποία παρέχει μία διεπαφή για τον προγραμματισμό σε μηχανήματα ενός κόμβου ή σε clusters με έμμεσο παραλληλισμό των δεδομένων και ανθεκτικότητα σε σφάλματα. Υποστηρίζει τη χρήση του μέσω αρκετών γλωσσών προγραμματισμού, όπως την Python, την Scala, την Java και άλλες. Πέρα από το βασικό του πυρήνα (Spark Core), ο οποίος είναι υπεύθυνος για την εκτέλεση των βασικών του λειτουργιών, υπάρχουν διάφορες επεκτάσεις. Ανάμεσα σε αυτές είναι το Spark Streaming, μέσω του οποίου μπορεί το Spark να χειρίζεται συνεχούς ροής δεδομένα σε mini-batches για την επεξεργασία και το μετασχηματισμό τους, ενώ επίσης υπάρχει και το Spark MLlib, το οποίο υποστηρίζει την εκτέλεση αλγορίθμων μηχανικής μάθησης. Τέλος, το Spark υποστηρίζει την επεξεργασία δεδομένων, τα οποία προέρχονται από διάφορες πηγές, όπως πληθώρα συστημάτων αρχείων (π.χ. HDFS, HBase), βάσεων δεδομένων ή ακόμα και το Apache Kafka.

Η επιλογή του Apache Spark, για να αποτελεί μέρος του τρέχοντος συστήματος συστάσεων, φαίνεται ιδανική μιας και δίνεται η δυνατότητα γρήγορης εκτέλεσης του αλγορίθμου ALS και άντλησης δεδομένων από το Apache Kafka. Ωστόσο, η επιλογή απορρίφθηκε, καθώς, όπως προέκυψε, η εκτέλεση του συγκεκριμένου αλγορίθμου της βιβλιοθήκης MLlib, με τη χρήση συνεχούς ροής δεδομένων, δεν υποστηρίζεται.

Με δεδομένους τους περιορισμούς του Spark και με στόχο την απλούστευση της αρχιτεκτονικής του συστήματος, ο σχεδιασμός του στράφηκε σε βιβλιοθήκες της γλώσσας Python. Πιο συγκεκριμένα, αρχικά εξετάστηκε το ενδεχόμενο να χρησιμοποιηθεί η βιβλιοθήκη κατασκευής και ανάλυσης συστημάτων συστάσεων Surprise. Ωστόσο, σύντομα διαπιστώθηκε ότι η συγκεκριμένη υποστηρίζει αποκλειστικά τη χρήση δεδομένων ρητής ανατροφοδότησης, κάτι το οποίο έρχεται σε αντίθεση με τη βασική σχεδιαστική επιλογή του να χρησιμοποιούνται δεδομένα έμμεσης ανατροφοδότησης. Επομένως, η χρήση του αυτού του εργαλείου απορρίφθηκε.

Αξίζει επίσης να αναφερθεί, ότι ο αρχικός σχεδιαστικός στόχος για το σύστημα συστάσεων ήταν η λειτουργία του με τη χρήση συνεργατικού φιλτραρίσματος με βάση τη μνήμη, αξιοποιώντας έναν neighborhood-based αλγόριθμο. Κάτι τέτοιο θα επέτρεπε την απλούστευση του τρόπου λειτουργίας του συστήματος, δίνοντας παράλληλα ικανοποιητικά αποτελέσματα, όσον αφορά την ακρίβεια των συστάσεων. Ωστόσο, δεδομένου ότι το σύστημα θα πρέπει να ανταποκρίνεται γρήγορα και να ενημερώνεται για τις αλλαγές στο περιεχόμενο της εφαρμογής και τις προτιμήσεις των χρηστών, η χρήση αυτής της μεθόδου δημιουργεί πρόβλημα, καθώς αυξάνεται ο όγκος των δεδομένων, διότι οι υπολογισμοί γίνονται ιδιαίτερα απαιτητικοί σε χρόνο και υπολογιστικούς πόρους. Έτσι, ο σχεδιασμός στράφηκε στη μέθοδο του συνεργατικού φιλτραρίσματος με βάση το μοντέλο, στην τεχνική μείωσης των διαστάσεων του πίνακα και τον χειρισμό των δεδομένων μέσω ενός κυλιόμενου παραθύρου, προκειμένου να δοθεί μία ουσιαστική λύση.

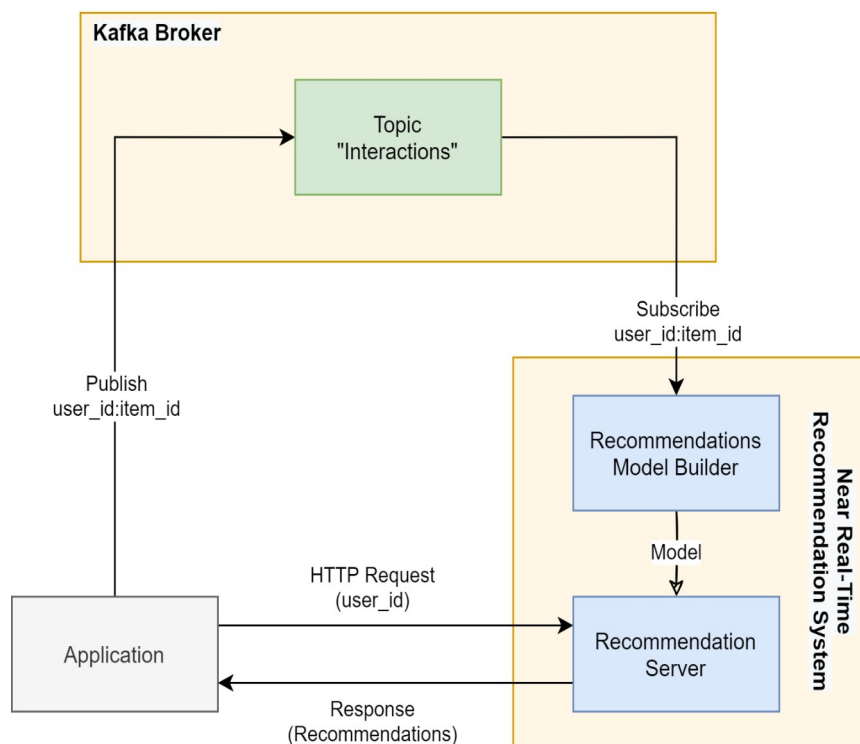
Κεφάλαιο 4

Υλοποίηση

ΣΕ αυτό το κεφάλαιο, παρουσιάζεται η αρχιτεκτονική του σχεδόν πραγματικού χρόνου συστήματος συστάσεων, όπως αυτή έχει προκύψει από τις σχεδιαστικές επιλογές του προηγούμενου κεφαλαίου. Έπειτα, περιγράφεται βήμα προς βήμα ο κώδικας, τόσο των βασικών προγραμμάτων του συστήματος, όσο και των βοηθητικών προγραμμάτων, ενώ παράλληλα επιδεικνύονται οι λειτουργίες τους. Τέλος, περιγράφονται οι τρόποι ενσωμάτωσης και χρήσης του συστήματος από μία εφαρμογή.

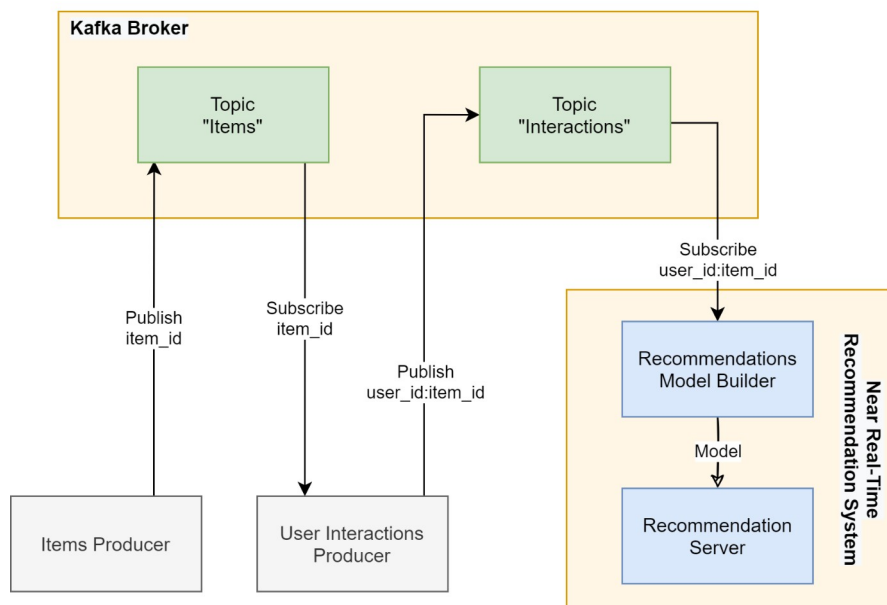
4.1 Αρχιτεκτονική του Συστήματος

Με βάση τις τελικές σχεδιαστικές επιλογές, οι οποίες ελήφθησαν στο προηγούμενο κεφάλαιο, προκύπτει ότι το σύστημα συστάσεων σχεδόν πραγματικού χρόνου απαρτίζεται από περισσότερα από ένα μέρη. Πιο συγκεκριμένα, υπάρχει το πρόγραμμα που αναλαμβάνει τη συλλογή των δεδομένων και την χρήση τους για την εκπαίδευση του μοντέλου συστάσεων, το πρόγραμμα, το οποίο δέχεται αιτήματα για παραγωγή συστάσεων, παράγει και επιστρέφει συστάσεις χρησιμοποιώντας το μοντέλο, ενώ παράλληλα υπάρχει εξυπηρετητής Kafka, που έχει το ρόλο μεταφοράς των συνεχούς ροής δεδομένων ανατροφοδότησης.



Εικόνα 4.1: Διάγραμμα Αρχιτεκτονικής του Συστήματος Συστάσεων

Όπως φαίνεται και στο διάγραμμα της αρχιτεκτονικής του συστήματος [Εικόνα 4.1], μία εφαρμογή (Application), η οποία χρησιμοποιεί το συγκεκριμένο σύστημα συστάσεων, προωθεί, κάθε φορά που προκύπτει μία αλληλεπίδραση χρήστη – αντικειμένου, το αναγνωριστικό του χρήστη (`user_id`) και το αναγνωριστικό του αντικειμένου (`item_id`) αυτού, σε ένα θέμα (Interactions) του εξυπηρετητή Kafka. Έπειτα, όντας εγγεγραμμένο στο συγκεκριμένο θέμα, το πρόγραμμα συλλογής των δεδομένων και εκπαίδευσης του μοντέλου (Recommendations Model Builder), λαμβάνει άμεσα κάθε νέα αλληλεπίδραση και μετά από έναν ορισμένο αριθμό νέων αλληλεπιδράσεων, εκπαιδεύει το μοντέλο συστάσεων. Επίσης, ορίζεται και το πλήθος των δεδομένων (αριθμός των πιο πρόσφατων αλληλεπιδράσεων), τα οποία συμμετέχουν στην εκπαίδευση. Όταν η εφαρμογή κρίνει απαραίτητο (ύστερα από αίτημα του χρήστη ή του συστήματος) να παρέχει σε ένα χρήστη συστάσεις, τότε στέλνει ένα αίτημα τύπου GET (HTTP Request) προς το πρόγραμμα παραγωγής συστάσεων (Recommendation Server), με παραμέτρους το αναγνωριστικό αυτού του χρήστη και το πλήθος των συστάσεων που πρέπει να επιστραφούν. Το πρόγραμμα λαμβάνοντας αυτό το αίτημα, υπολογίζει τις καλύτερες συστάσεις για το συγκεκριμένο χρήστη, με βάση το πιο πρόσφατο μοντέλο και επιστρέφει μία λίστα με τα αναγνωριστικά των αντικειμένων που προτείνονται.



Εικόνα 4.2: Προσθήκη Βοηθητικών Προγραμμάτων

Ωστόσο, κατά τη διάρκεια της ανάπτυξης, προκειμένου να διαπιστώνεται ότι το σύστημα λειτουργεί χωρίς προβλήματα και ότι δεν περιέχει σφάλματα, είναι απαραίτητο να εκτελείται δοκιμαστικά, σε συνθήκες αντίστοιχες με αυτές που θα δούλευε με μία πραγματική εφαρμογή. Για τον σκοπό αυτό, αναπτύχθηκαν άλλα δύο προγράμματα [Εικόνα 4.2], τα οποία έχουν βοηθητικό ρόλο και προσομοιώνουν το περιβάλλον (αντικείμενα, χρήστες, αλληλεπιδράσεις) μιας εφαρμογής. Το πρώτο πρόγραμμα (Item Producer), παράγει έναν ορισμένο αριθμό τυχαίων αναγνωριστικών αντικειμένων, τα οποία για πρακτικούς λόγους προωθούνται σε ένα θέμα (Items) του Kafka. Στη συνέχεια, το δεύτερο πρόγραμμα (User Interactions Producer) λαμβάνει από το ίδιο θέμα τα αναγνωριστικά των αντικειμένων και αφού έχει δημιουργήσει έναν ορισμένο αριθμό χρηστών (τυχαίων αναγνωριστικών χρηστών), παρασκευάζει αλληλεπιδράσεις χρηστών – αντικειμένων, αντιστοιχίζοντας με τυχαίο τρόπο χρήστες και αντικείμενα. Πρέπει να αναφερθεί ότι, ο λόγος για τον οποίο τα δύο προγράμματα είναι ανεξάρτητα, είναι ότι έτσι, μπορεί να επιτυγχάνεται ταυτόχρονη παραγωγή νέων αντικειμένων και αλληλεπιδράσεων, προσομοιώνοντας ακόμα καλύτερα τις συνθήκες μιας πραγματικής εφαρμογής.

4.2 Ανάλυση Κώδικα

Ο τρόπος, με τον οποίο κάθε ένα από τα προγράμματα, που αναφέρθηκαν παραπάνω, επιτυγχάνει το στόχο του, διατυπώνεται καλύτερα αναλύοντας τον πηγαίο κώδικα τους. Ο κώδικας, τόσο των βοηθητικών όσο και των βασικών προγραμμάτων βρίσκεται αναρτημένος σε ένα αποθετήριο στο GitHub [37], διαθέσιμος για ελεύθερη χρήση και συνεισφορά.

4.2.1 Πρόγραμμα Παραγωγής Αντικειμένων

Αρχικά, το βοηθητικό πρόγραμμα παραγωγής αντικειμένων, μέσω ενός μενού, ζητάει από τον χρήστη να επιλέξει αν η παραγωγή των αντικειμένων θα γίνει εφάπαξ, ζητώντας τον αριθμό των αντικειμένων που πρόκειται να παραχθούν, ή αν θα γίνεται επαναληπτικά με αυτόματο τρόπο. Στην πρώτη περίπτωση, η συνάρτηση παραγωγής αντικειμένων (`publish_item`) καλείται μία φορά, ορίζοντας τον αριθμό των αντικειμένων, ενώ στη δεύτερη, η συνάρτηση καλείται επαναληπτικά παράγοντας ένα κάθε φορά αντικείμενο.

```
while True:
    selection = input("\n\n1. Manual\n2. Auto\n\n> ")

    if selection == '1':
        items_num = input('Number of items: ')
        publish_item(items_topic, key, int(items_num))

    if selection == '2':
        while True:
            publish_item(items_topic, key, 1)
            time.sleep(round(random.uniform(5,10),1))
```

Απόσπασμα Κώδικα 4.1: Μενού του Προγράμματος Παραγωγής Αντικειμένων

Κατά την κλήση της συνάρτησης παραγωγής αντικειμένων, για κάθε αντικείμενο που πρόκειται να παραχθεί (το πλήθος ορίζεται κατά την κλήση), επιλέγεται τυχαία μία τιμή, ως αναγνωριστικό αντικειμένου και προωθείται στο αντίστοιχο θέμα, αφού έχει προηγηθεί έλεγχος για το αν η συγκεκριμένη τιμή υπάρχει ήδη στο θέμα. Στην τρέχουσα έκδοση του προγράμματος, κάθε τιμή προκύπτει επιλέγοντας τυχαία μία εγγραφή, ενός αρχείου με δεδομένα.

```
news = file.readlines()
for i in range(0, number_of_items):
    while True:
        item_id = bytes(json.loads(random.choice(news))['id'], encoding='utf-8')
        if item_id not in existing_items:
            break
    producer.send(topic_name, key=key, value=item_id)
```

Απόσπασμα Κώδικα 4.2: Παραγωγή Τυχαίων Αναγνωριστικών Αντικειμένων

4.2.2 Πρόγραμμα Παραγωγής Αλληλεπιδράσεων Χρήστη - Αντικειμένου

Αντίστοιχα, το βοηθητικό πρόγραμμα παραγωγής αλληλεπιδράσεων χρήστη – αντικειμένου, μέσω ενός μενού, ζητάει από τον χρήστη να επιλέξει ανάμεσα στην παραγωγή χρηστών και στην παραγωγή αλληλεπιδράσεων.

```

while True:
    selection = input("\n\n1. Create users\n2. Create interactions\n\n> ")

    if selection == '1':
        users_number = input('Number of users: ')
        for user in range(int(users_number)):
            users.append(User(users))

    if selection == '2':
        if not users:
            print('Create some users first\n')
        else:
            while True:
                random.choice(users).create_interaction(items, key,
                                                         interactions_topic)

                time.sleep(round(random.uniform(0,3),1))

```

Απόσπασμα Κώδικα 4.3: Μενού του Προγράμματος Παραγωγής Αλληλεπιδράσεων

Επιλέγοντας, την παραγωγή χρηστών, το πρόγραμμα ζητάει τον αριθμό των χρηστών που πρόκειται να παραχθούν και για κάθε έναν από αυτούς δημιουργείται ένα αντικείμενο της κλάσης “User”. Κατά τη δημιουργία κάθε αντικειμένου ανατίθεται μία τυχαία τιμή ως το αναγνωριστικό χρήστη.

```

def __init__(self, previous_users):
    while True:
        user_id = random.randint(0, 1000)
        if user_id not in previous_users:
            self.user_id = user_id
            break

```

Απόσπασμα Κώδικα 4.4: Ανάθεση Τυχαίων Μοναδικών Αναγνωριστικών στους Χρήστες

Από την άλλη, επιλέγοντας στο μενού την παραγωγή αλληλεπιδράσεων, καλείται επ’ αόριστον η συνάρτηση παραγωγής αλληλεπιδράσεων (create_interactions) για έναν τυχαίο κάθε φορά χρήστη. Η συνάρτηση επιλέγει τυχαία ένα από τα αναγνωριστικά αντικειμένων, τα οποία λήφθηκαν νωρίτερα από το θέμα, και το συνενώνει με το αναγνωριστικό του τρέχοντος χρήστη. Τέλος, η αλληλεπίδραση που παράχθηκε προωθείται στο αντίστοιχο θέμα (Interactions) στο Kafka.

```

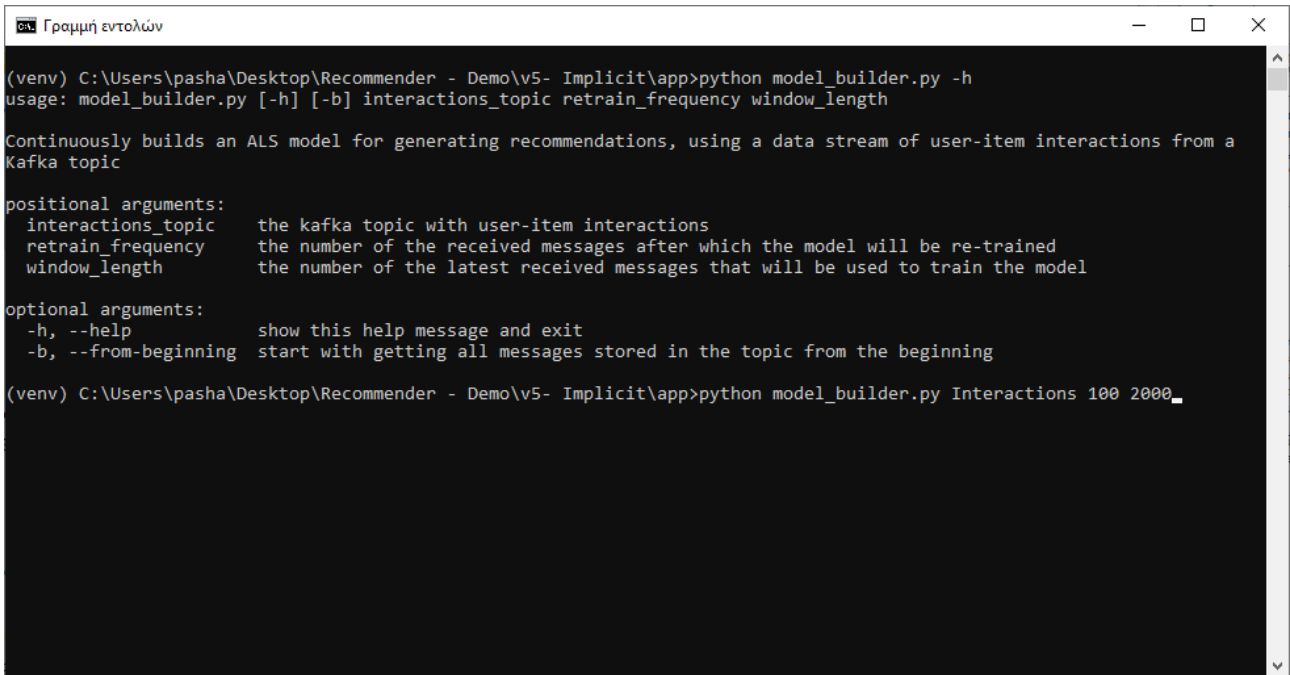
def create_interaction(self, items, key, interactions_topic):
    item = random.choice(items)
    interaction = bytes(f'{self.user_id}:{item}', encoding='utf-8')
    producer.send(interactions_topic, interaction, key)

```

Απόσπασμα Κώδικα 4.5: Παραγωγή Τυχαίων Αλληλεπιδράσεων

4.2.3 Πρόγραμμα Εκπαίδευσης Μοντέλου

Όπως αναφέρθηκε προηγουμένως, το πρόγραμμα εκπαίδευσης μοντέλου είναι υπεύθυνο για τη λήψη των συνεχούς ροής δεδομένων και την αξιοποίησή τους στην εκπαίδευση του μοντέλου συστάσεων. Η εκτέλεση αυτού του προγράμματος πραγματοποιείται μόνο εφόσον έχουν δοθεί τιμές για τις παραμέτρους, “interactions_topic”, “retrain_frequency” και “window_length” [Εικόνα 4.3]. Η πρώτη παράμετρος αποτελεί το όνομα του θέματος, το οποίο περιλαμβάνει τις αλληλεπιδράσεις. Η τιμή της δεύτερης παραμέτρου αντιπροσωπεύει τη συχνότητα με την οποία θα γίνεται εκ νέου εκπαίδευση του μοντέλου, δηλαδή το πλήθος νέων αλληλεπιδράσεων που πρέπει να ληφθούν ώστε να παραχθεί ένα νέο μοντέλο, ενώ η τρίτη παράμετρος καθορίζει το πλήθος των πιο πρόσφατων αλληλεπιδράσεων που θα συμπεριλαμβάνονται στην εκπαίδευση.



```

(venv) C:\Users\pasha\Desktop\Recommender - Demo\v5- Implicit\app>python model_builder.py -h
usage: model_builder.py [-h] [-b] interactions_topic retrain_frequency window_length

Continuously builds an ALS model for generating recommendations, using a data stream of user-item interactions from a
Kafka topic

positional arguments:
  interactions_topic  the kafka topic with user-item interactions
  retrain_frequency  the number of the received messages after which the model will be re-trained
  window_length      the number of the latest received messages that will be used to train the model

optional arguments:
  -h, --help            show this help message and exit
  -b, --from-beginning  start with getting all messages stored in the topic from the beginning

(venv) C:\Users\pasha\Desktop\Recommender - Demo\v5- Implicit\app>python model_builder.py Interactions 100 2000_

```

Εικόνα 4.3: Παράμετροι του Προγράμματος Εκπαίδευσης του Μοντέλου

Έχοντας ορίσει τιμές για αυτές τις παραμέτρους, το πρόγραμμα ξεκινάει να συλλέγει τα νέα δεδομένα που μπαίνουν στο θέμα των αλληλεπιδράσεων. Σε πρώτο στάδιο, τα δεδομένα τοποθετούνται σε μία FIFO ουρά, με μέγιστο μέγεθος τον αριθμό του παραθύρου, που δόθηκε ως παράμετρος. Με αυτό τον τρόπο, τίθεται όριο στο πλήθος των δεδομένων που διατηρούνται, αλλά επίσης διατηρούνται μόνο τα πιο πρόσφατα δεδομένα.

```

q = queue.Queue(maxsize=window_length) # A FIFO queue
for msg in consumer:
    if q.full():
        q.get()
        q.put(msg)
    else:
        q.put(msg)

```

Απόσπασμα Κώδικα 4.6: Κυλιόμενο Παράθυρο Αλληλεπιδράσεων με τη Μορφή Ουράς

Μετρώντας τον αριθμό των δεδομένων που λαμβάνονται, η διαδικασία προετοιμασίας των δεδομένων και εκπαίδευσης του μοντέλου ξεκινάει μόλις ο αριθμός των ληφθέντων αλληλεπιδράσεων είναι ίσος με την ορισμένη συχνότητα εκπαίδευσης. Έτσι, όταν συμβεί αυτό, χρησιμοποιώντας όλες τις αλληλεπιδράσεις που βρίσκονται στην ουρά, γίνεται μία καταμέτρηση των εμφανίσεων κάθε μοναδικής αλληλεπίδρασης. Στη συνέχεια τα δεδομένα μετατρέπονται σε κατάλληλη μορφή (pandas dataframe).

```

if msg_count == freq:
    counts = dict()
    users = []
    items = []
    interactions = []
    for lmsg in q.queue:
        # Count the occurrences of each interaction
        if lmsg in counts:
            counts[lmsg] += 1
        else:
            counts[lmsg] = 1
        users.append(int(lmsg.split(':')[0]))
        items.append(int(lmsg.split(':')[1]))
        interactions.append(counts[lmsg])
    df = pd.DataFrame(list(zip(users, items, interactions)),
                      columns=['User', 'Item', 'Interaction'])

```

Απόσπασμα Κώδικα 4.7: Προετοιμασία των Δεδομένων για την Εκπαίδευση του Μοντέλου

Προκειμένου το πρόγραμμα να συνεχίσει να λαμβάνει αλληλεπιδράσεις και κατά την διάρκεια εκπαίδευσης του μοντέλου συστάσεων, ορίζεται η εκτέλεση της συνάρτησης κατασκευής του μοντέλου (`create_recommendations_model`) να γίνεται παράλληλα, σε ξεχωριστό thread. Με την έναρξη της κατασκευής του μοντέλου, ταυτόχρονα μηδενίζεται και ο μετρητής λήψης νέων αλληλεπιδράσεων.

```

# if there is no active thread for training
if init or not t1.is_alive():
    users_col = df['User']
    items_col = df['Item']
    interactions_col = df['Interaction']

    # Retrain the model on a different thread
    t1 = threading.Thread(target=create_recommendations_model,
                        args=(users_col, items_col, interactions_col))

    t1.start()
    init = False
msg_count = 0

```

Απόσπασμα Κώδικα 4.8: Κατασκευή Μοντέλου σε Διαφορετικό Thread

Η βιβλιοθήκη, η οποία παρέχει την υλοποίηση του αλγορίθμου παραγωντοποίησης πίνακα, που θα χρησιμοποιηθεί, απαιτεί τα δεδομένα να έχουν την μορφή ενός πίνακα χρήστη αντικειμένου. Επομένως, δημιουργούνται δύο πίνακες, ένας πίνακας αντικειμένου – χρήστη, για την εκπαίδευση του μοντέλου και ένας πίνακας χρήστη – αντικειμένου (αντιμετάθεση του πρώτου) για την παραγωγή των συστάσεων [33]. Ο δεύτερος πίνακας αποθηκεύεται ως αρχείο, καθώς θα χρησιμοποιείται ανεξάρτητα, όταν θα γίνονται αιτήματα παραγωγής συστάσεων.

```

def create_recommendations_model(users_column, items_column, interactions_column):
    # Create one matrix for fitting the model (item-user)
    sparse_item_user = sparse.csr_matrix((interactions_column.astype(float),
    (items_column, users_column)))

    # Transposing the item-user matrix to create a matrix for recommendations
    # (user-item)
    sparse_user_item = sparse_item_user.T.tocsr()

    # Save the user-item matrix for making recommendations asynchronously
    save_npz("sparse_user_item.npz", sparse_user_item)

```

Απόσπασμα Κώδικα 4.9: Μετατροπή των Δεδομένων σε Πίνακες Χρήστη - Αντικειμένου

Τέλος, ορίζονται οι παράμετροι του αλγορίθμου ALS, δηλαδή ο αριθμός των κρυφών παραγόντων, η τιμή της παραμέτρου κανονικοποίησης και ο αριθμός των επαναλήψεων, υπολογίζεται η εμπιστοσύνη που εμπεριέχεται στα δεδομένα [7] και γίνεται η εκπαίδευση του μοντέλου.

```
# Initialize the als model
model = implicit.als.AlternatingLeastSquares(factors=20, regularization=0.1,
                                              iterations=20)

# Calculate the confidence by multiplying by alpha value.
alpha_val = 15
data_conf = (sparse_item_user * alpha_val).astype('double')

# Fit the model
model.fit(data_conf)

# Serialize and save the model
with open('model.sav', 'wb') as pickle_out:
    pickle.dump(model, pickle_out)
```

Απόσπασμα Κώδικα 4.10: Εκπαίδευση του Μοντέλου με τον Αλγόριθμο ALS

Το μοντέλο αποθηκεύεται σε δυαδική μορφή ώστε να μπορεί να χρησιμοποιηθεί αργότερα για την παραγωγή συστάσεων.

4.2.4 Πρόγραμμα Παροχής Συστάσεων

Το πρόγραμμα παροχής συστάσεων είναι στην πράξη ένας εξυπηρετητής δικτύου, ο οποίος λειτουργεί χρησιμοποιώντας το πρωτόκολλο HTTP. Αναλυτικότερα, δέχεται αιτήματα τύπου GET, με παραμέτρους το αναγνωριστικό ενός χρήστη (userId), για τον οποίο προορίζονται οι συστάσεις και τον αριθμό αυτών των συστάσεων (recommendationsNumber) που πρόκειται να παραχθούν. Έτσι, στην πράξη, με την λήψη ενός τέτοιου αιτήματος καλείται η συνάρτηση (make_recommendations), που αναλαμβάνει τον υπολογισμό τους. Το πρόγραμμα θα μπορούσε να χαρακτηριστεί και ως ένας εξυπηρετητής συστάσεων.

```
@app.route('/', methods=['GET'])
def send_recommendations():
    try:
        if request.args.get('userId') is not None
           and request.args.get('recommendationsNumber') is not None:
            result = make_recommendations(int(request.args.get('userId')),
                                           int(request.args.get('recommendationsNumber')))
            if result == -1:
                raise ValueError
            else:
                return result
        else:
            raise ValueError
    except ValueError:
        abort(400, 'Bad Request\nCheck Parameters')
```

Απόσπασμα Κώδικα 4.11: Λήψη Αιτημάτων GET για Παραγωγή Συστάσεων

Για τον υπολογισμό των συστάσεων, το πρόγραμμα αρχικά πραγματοποιεί ανάκτηση του πίνακα χρήστη – αντικειμένου, ο οποίος παράχθηκε τελευταίος, και του πιο πρόσφατου μοντέλου συστάσεων. Τέλος, κάνοντας χρήση του μοντέλου υπολογίζονται συστάσεις για τον επιθυμητό χρήστη και επιστρέφονται, με την κατάλληλη μορφή [Εικόνα 4.4], αυτές που κρίνονται ως οι πιο ακριβείς, σύμφωνα πάντα και με το επιθυμητό πλήθος τους.

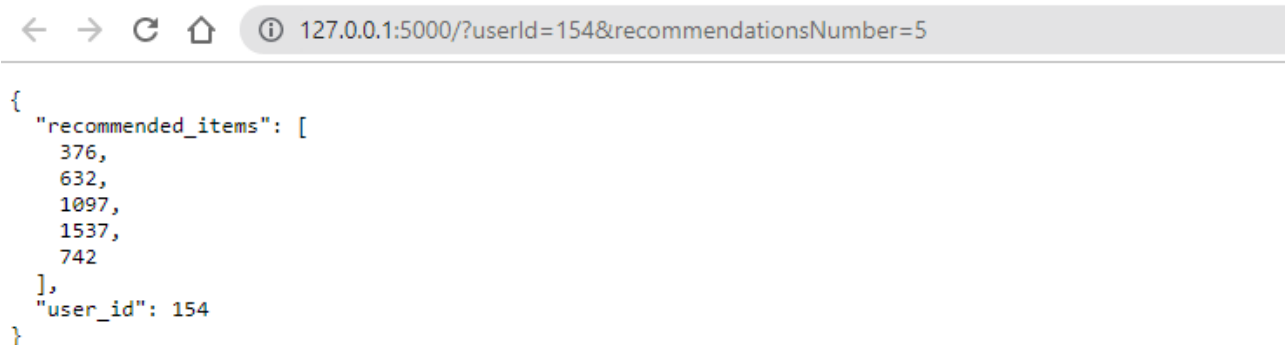
```
def make_recommendations(target_user_id, n):
    # Load the stored user-item matrix
    sparse_user_item = load_npz("sparse_user_item.npz")

    # Load the stored recommendations model
    with open('model.sav', 'rb') as pickle_in:
        model = pickle.load(pickle_in)

    try:
        # Make recommendations using the model
        recommended = model.recommend(target_user_id, sparse_user_item, n,
                                       filter_already_liked_items=True)
    except IndexError:
        return -1

    recommended_items = []
    for item in recommended:
        recommended_items.append(int(item[0]))
    return {'user_id': target_user_id,
            'recommended_items': recommended_items}
```

Απόσπασμα Κώδικα 4.12: Παραγωγή Συνστάσεων με Βάση το Τελευταίο Μοντέλο



```
{
  "recommended_items": [
    376,
    632,
    1097,
    1537,
    742
  ],
  "user_id": 154
}
```

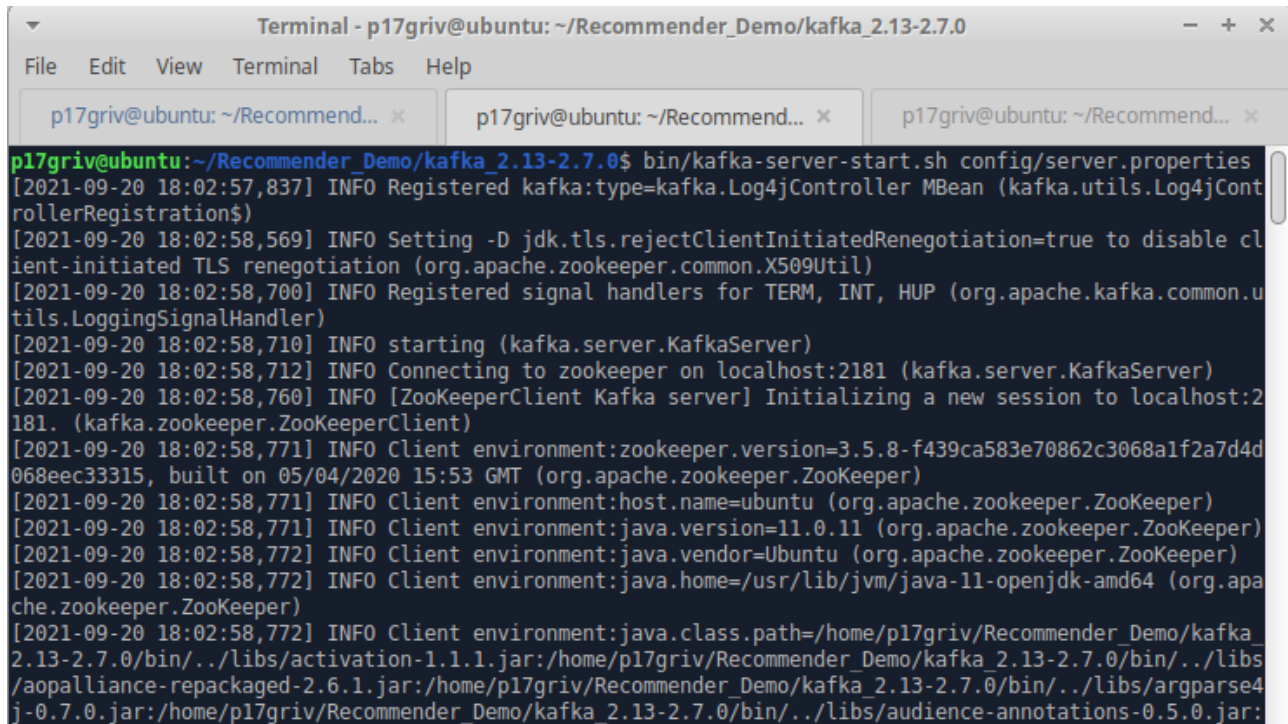
Εικόνα 4.4: Επιστροφή 5 Προτεινόμενων Αντικειμένων για τον Χρήστη “154”

4.3 Ενσωμάτωση του Συστήματος

Έχοντας περιγράψει τον κώδικα των προγραμμάτων του συστήματος, η περιγραφή του τρόπου με τον οποίο το σύστημα μπορεί να ενσωματωθεί και να χρησιμοποιηθεί από μία εφαρμογή, είναι επακόλουθη. Αρχικά, ο προγραμματιστής μιας εφαρμογής, που απαιτεί την παροχή ποιοτικών και διαρκώς ενημερωμένων συστάσεων στους χρήστες της, ορίζει την ερμηνεία της αλληλεπίδρασης χρήστη - αντικείμενου. Αλληλεπίδραση στην εφαρμογή μπορεί να προκύπτει με απλό τρόπο από μία ενέργεια του χρήστη, όπως κλικ σε ένα αντικείμενο (π.χ. προβολή της σελίδας ενός προϊόντος), ή μέσω μιας συνθήκης, η οποία ελέγχει αν μια ενέργεια του χρήστη με ένα αντικείμενο, μπορεί να θεωρηθεί αλληλεπίδραση (π.χ. έλεγχος για το αν ο χρόνος παραμονής ενός χρήστη σε ένα άρθρο ξεπερνά ένα κατώτατο όριο).

Έπειτα, ο προγραμματιστής της εφαρμογής, κάνοντας ορισμένες παρεμβάσεις στον κώδικά της, δηλώνει την αποστολή (publish) των αλληλεπιδράσεων των χρηστών, όταν αυτές προκύπτουν. Επομένως, εν συνεχεία οφείλει να εγκαταστήσει μία έκδοση του Apache Kafka, προκειμένου να μπορούν τα δεδομένα να προωθούνται σε ένα topic ενός broker. Αυτός ο εξυπηρετητής Kafka μπορεί να εγκατασταθεί είτε τοπικά,

στο ίδιο μηχάνημα που φιλοξενεί και τον εξυπηρετητή της εφαρμογής, είτε σε ένα άλλο φυσικό μηχάνημα που μπορεί να διαθέτει ή σε μηχάνημα μίας cloud υπηρεσίας, που ενδεχομένως να αξιοποιείται. Όπως προαναφέρθηκε, η λειτουργία του Kafka προϋποθέτει την ύπαρξη του Apache ZooKeeper εγκατεστημένου, έτσι ώστε να εκτελείται παράλληλα.



```

p17griv@ubuntu: ~/Recommender_Demo/kafka_2.13-2.7.0$ bin/kafka-server-start.sh config/server.properties
[2021-09-20 18:02:57,837] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2021-09-20 18:02:58,569] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2021-09-20 18:02:58,700] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2021-09-20 18:02:58,710] INFO starting (kafka.server.KafkaServer)
[2021-09-20 18:02:58,712] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2021-09-20 18:02:58,760] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2021-09-20 18:02:58,771] INFO Client environment:zookeeper.version=3.5.8-f439ca583e70862c3068a1f2a7d4d068eec33315, built on 05/04/2020 15:53 GMT (org.apache.zookeeper.ZooKeeper)
[2021-09-20 18:02:58,771] INFO Client environment:host.name=ubuntu (org.apache.zookeeper.ZooKeeper)
[2021-09-20 18:02:58,771] INFO Client environment:java.version=11.0.11 (org.apache.zookeeper.ZooKeeper)
[2021-09-20 18:02:58,772] INFO Client environment:java.vendor=Ubuntu (org.apache.zookeeper.ZooKeeper)
[2021-09-20 18:02:58,772] INFO Client environment:java.home=/usr/lib/jvm/java-11-openjdk-amd64 (org.apache.zookeeper.ZooKeeper)
[2021-09-20 18:02:58,772] INFO Client environment:java.class.path=/home/p17griv/Recommender_Demo/kafka_2.13-2.7.0/bin/../libs/activation-1.1.1.jar:/home/p17griv/Recommender_Demo/kafka_2.13-2.7.0/bin/../libs/aopalliance-repackaged-2.6.1.jar:/home/p17griv/Recommender_Demo/kafka_2.13-2.7.0/bin/../libs/argparse4j-0.7.0.jar:/home/p17griv/Recommender_Demo/kafka_2.13-2.7.0/bin/../libs/audience-annotations-0.5.0.jar:
  
```

Εικόνα 4.5: Εκκίνηση του Εξυπηρετητή Kafka

Σε ότι αφορά τα (βασικά) προγράμματα του συστήματος συστάσεων, αυτά μπορούν αντίστοιχα να εκτελούνται σε ένα μηχάνημα της επιλογής του προγραμματιστή. Ωστόσο, για το λόγο ότι το σύστημα πραγματοποιεί απαιτητικούς υπολογισμούς, ιδίως όταν έχει επιλεχθεί η εκπαίδευση του μοντέλου να γίνεται με πολλά δεδομένα, προτείνεται η εκτέλεση τους να γίνεται σε ξεχωριστό υπολογιστικό σύστημα, από αυτό στο οποίο φιλοξενείται ο εξυπηρετητής της εφαρμογής. Επιπλέον, κατά την έναρξη της εκτέλεσης των προγραμμάτων, όπως φάνηκε και από την ανάλυση του κώδικα, ο προγραμματιστής καλείται να δώσει τιμές σε ορισμένες παραμέτρους, όπως το πλήθος των δεδομένων εκπαίδευσης και τη συχνότητα επανεκπαίδευσης του μοντέλου. Οι κατάλληλες τιμές αυτών των παραμέτρων μπορούν να βρεθούν είτε επιβλέποντας τις επιδόσεις του υπολογιστικού συστήματος, έτσι ώστε οι υπολογισμοί να πραγματοποιούνται σε ένα εύλογο χρονικό διάστημα, ή διεξάγοντας έρευνα για την ποιότητα των συστάσεων, όπως περιγράφεται στο επόμενο κεφάλαιο, αυτό της αξιολόγησης.

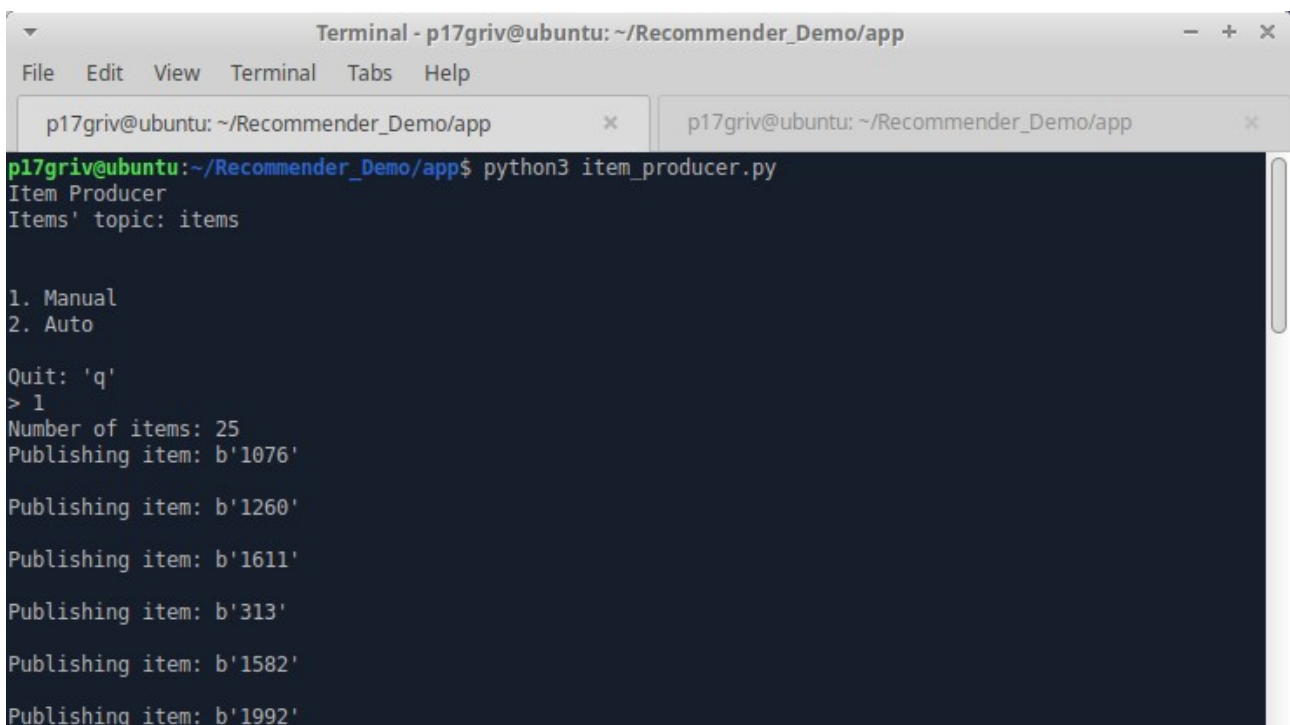
Οι τελευταίες ενέργειες του προγραμματιστή σχετίζονται με το κομμάτι της παροχής συστάσεων. Έτσι, με παρέμβαση στον κώδικα της εφαρμογής, ορίζεται το πότε θα γίνεται η αποστολή αιτημάτων παροχής συστάσεων προς τον εξυπηρετητή συστάσεων του συστήματος. Η εφαρμογή μπορεί να στέλνει αυτά τα αιτήματα, είτε επειδή το ορίζει ο επιθυμητός τρόπος λειτουργίας της, είτε κατόπιν απαίτησης των ιδίων των χρηστών της. Στη συνέχεια, έχοντας λάβει την απάντηση με τα προτεινόμενα αντικείμενα, η οποία έχει μία βολική μορφή τύπου εγγράφου JSON [Εικόνα 4.4], ο προγραμματιστής έχει τη δυνατότητα να τα προβάλλει στον χρήστη, με όποιον τρόπο κρίνει κατάλληλο.

4.4 Δοκιμή του Συστήματος

Όπως αναφέρθηκε προηγουμένως στις ενότητες περιγραφής της αρχιτεκτονικής του συστήματος και της ανάλυσης του κώδικα των προγραμμάτων του, μπορεί να πραγματοποιηθεί δοκιμή της λειτουργίας του συστήματος δίχως την ύπαρξη και τη διασύνδεση μιας πραγματικής εφαρμογής, χρησιμοποιώντας τα

βοηθητικά προγράμματα. Τα προγράμματα αυτά είναι σε θέση να προσομοιώνουν τις συνθήκες μιας εφαρμογής σε βαθμό όπου συμπεραίνεται η λειτουργικότητα του συστήματος αλλά όχι η αποτελεσματικότητά του, καθώς λειτουργούν με τυχαίο τρόπο.

Αναλυτικότερα, το πρόγραμμα παραγωγής αντικειμένων, δημιουργεί αντικείμενα, παράγοντας με τυχαίο τρόπο αναγνωριστικά αντικειμένων. Αυτό μπορεί να γίνει είτε μία φορά, αντιπροσωπεύοντας την περίπτωση μιας εφαρμογής με σταθερό περιεχόμενο, ή επαναλαμβανόμενα ανά τυχαία (σχετικά μικρά για πρακτικούς λόγους) χρονικά διαστήματα, στην περίπτωση μιας εφαρμογής, της οποίας το περιεχόμενο αυξάνεται δυναμικά. Η επιλογή ανάμεσα σε αυτούς τους δύο τρόπους λειτουργίας μπορεί να γίνει μέσω ενός απλού μενού, αμέσως μετά την εκτέλεση του προγράμματος, ενώ στην πρώτη περίπτωση ζητείται επιπλέον και ο επιθυμητός αριθμός των χρηστών [Εικόνα 4.6]. Πρέπει να σημειωθεί ότι η εκτέλεση του βοηθητικού προγράμματος παραγωγής αντικειμένων πρέπει να γίνει προηγείται της εκτέλεσης του προγράμματος παραγωγής αλληλεπιδράσεων στην περίπτωση όπου το σύστημα δοκιμάζεται για πρώτη φορά και δεν υπάρχει κανένα αντικείμενο στο αντίστοιχο θέμα στον εξυπηρετητή Kafka.



```
Terminal - p17griv@ubuntu: ~/Recommender_Demo/app
File Edit View Terminal Tabs Help
p17griv@ubuntu: ~/Recommender_Demo/app x p17griv@ubuntu: ~/Recommender_Demo/app x
p17griv@ubuntu:~/Recommender_Demo/app$ python3 item_producer.py
Item Producer
Items' topic: items

1. Manual
2. Auto

Quit: 'q'
> 1
Number of items: 25
Publishing item: b'1076'

Publishing item: b'1260'

Publishing item: b'1611'

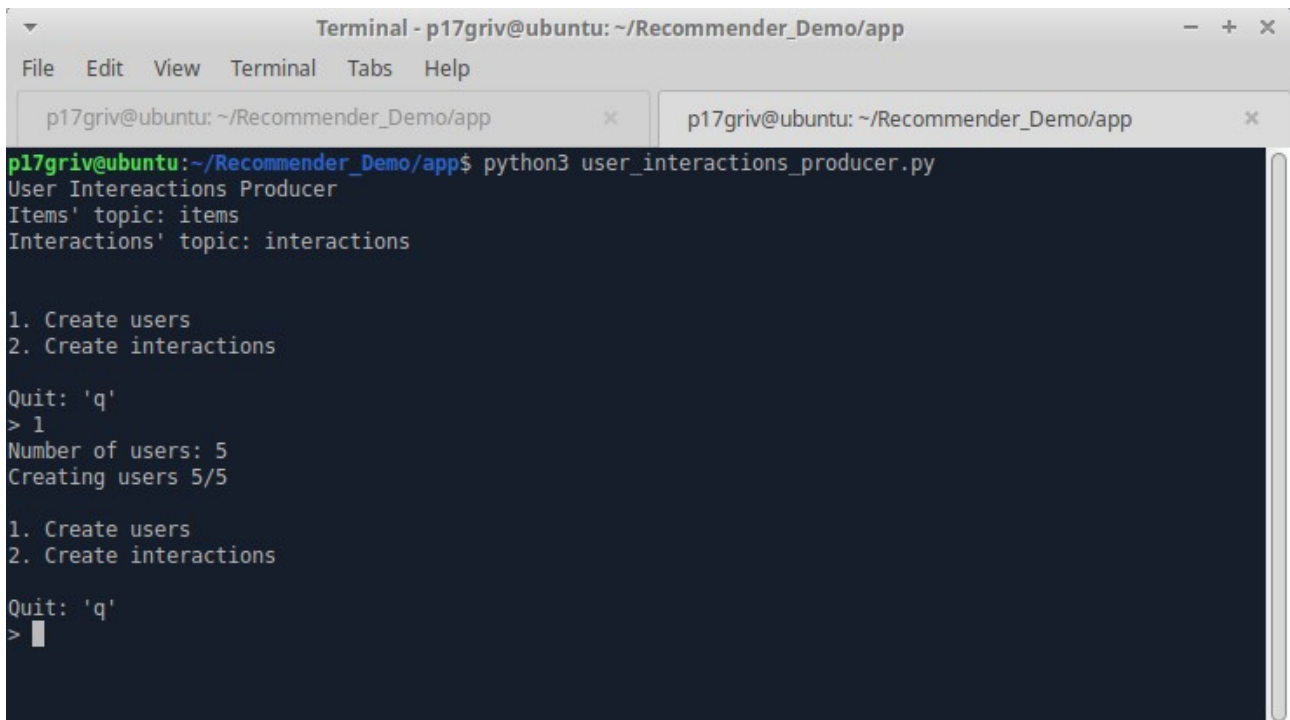
Publishing item: b'313'

Publishing item: b'1582'

Publishing item: b'1992'
```

Εικόνα 4.6: Δοκιμή του Συστήματος - Παραγωγή 25 Αντικειμένων

Από την άλλη, το βοηθητικό πρόγραμμα παραγωγής αλληλεπιδράσεων χρηστών με αντικείμενα, αρχικά παράγει με τυχαίο τρόπο χρήστες, ζητώντας το επιθυμητό πλήθος τους [Εικόνα 4.7], και στη συνέχεια αντιστοιχίζει επίσης με τυχαίο τρόπο τους χρήστες αυτούς με τα αντικείμενα που δημιουργήθηκαν νωρίτερα. Με άλλα λόγια το πρόγραμμα παράγει τις αλληλεπιδράσεις των χρηστών με αντικείμενα, οι οποίες θα προέκυπταν σε μία πραγματική εφαρμογή. Ωστόσο, βασική διαφορά είναι το γεγονός ότι οι αλληλεπιδράσεις παράγονται με τυχαίο τρόπο, πράγμα που είναι ο βασικός λόγος για τον οποίο δεν μπορεί να αξιολογηθεί η αποτελεσματικότητα του συστήματος.



```
Terminal - p17griv@ubuntu: ~/Recommender_Demo/app
File Edit View Terminal Tabs Help
p17griv@ubuntu: ~/Recommender_Demo/app x p17griv@ubuntu: ~/Recommender_Demo/app x
p17griv@ubuntu:~/Recommender_Demo/app$ python3 user_interactions_producer.py
User Intereactions Producer
Items' topic: items
Interactions' topic: interactions

1. Create users
2. Create interactions

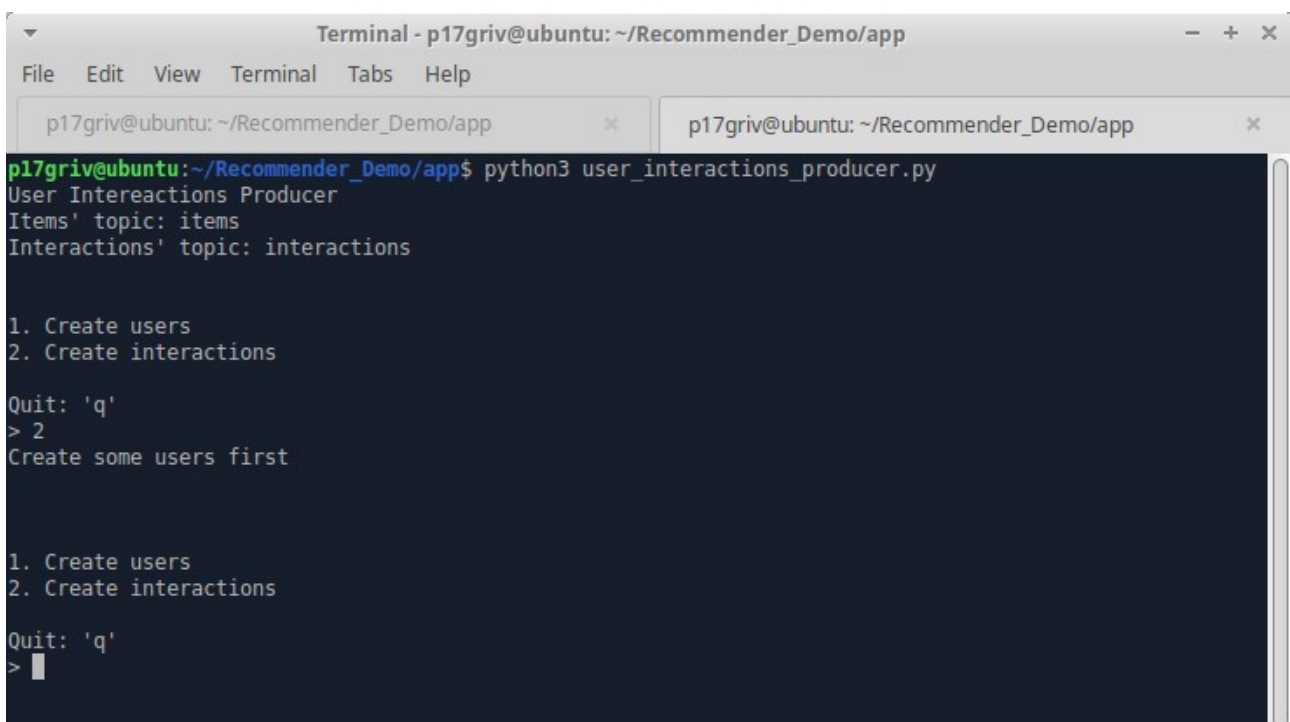
Quit: 'q'
> 1
Number of users: 5
Creating users 5/5

1. Create users
2. Create interactions

Quit: 'q'
> █
```

Εικόνα 4.7: Δοκιμή του Συστήματος - Παραγωγή 5 Χρηστών

Παρόμοια με το βοηθητικό πρόγραμμα παραγωγής αντικειμένων, ο χειρισμός του προγράμματος γίνεται μέσω ενός απλού μενού, από το οποίο μπορεί να επιλεγθεί η παραγωγή χρηστών και η παραγωγή αλληλεπιδράσεων. Στο σημείο αυτό, πρέπει να αναφερθεί ότι η παραγωγή χρηστών θα πρέπει να προηγείται της παραγωγής αλληλεπιδράσεων καθώς σε αντίθετη περίπτωση δεν είναι δυνατή η σύνδεση των αντικειμένων με χρήστες. Αυτό το λογικό σφάλμα χρήσης αποτρέπεται από το πρόγραμμα, εμφανίζοντας παράλληλα κατάλληλο μήνυμα λάθους [Εικόνα 4.8].



```
Terminal - p17griv@ubuntu: ~/Recommender_Demo/app
File Edit View Terminal Tabs Help
p17griv@ubuntu: ~/Recommender_Demo/app x p17griv@ubuntu: ~/Recommender_Demo/app x
p17griv@ubuntu:~/Recommender_Demo/app$ python3 user_interactions_producer.py
User Intereactions Producer
Items' topic: items
Interactions' topic: interactions

1. Create users
2. Create interactions

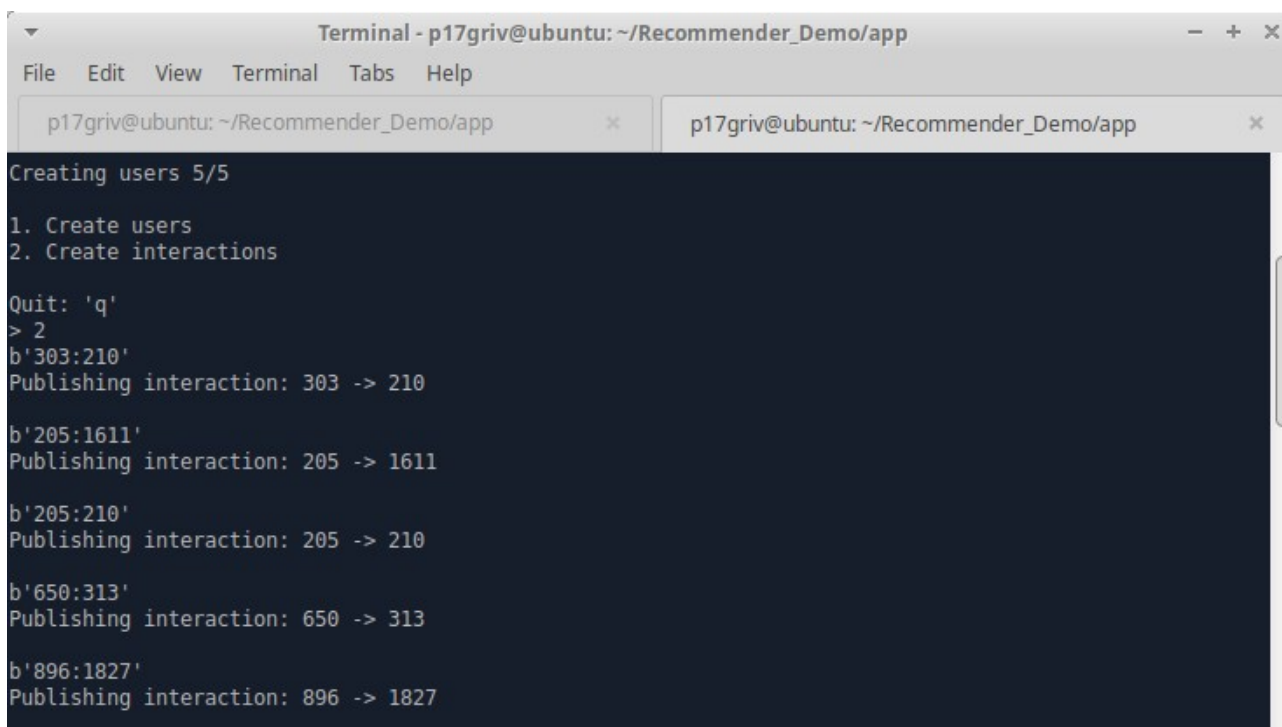
Quit: 'q'
> 2
Create some users first

1. Create users
2. Create interactions

Quit: 'q'
> █
```

Εικόνα 4.8: Σφάλμα Χρήσης Βοηθητικού Προγράμματος

Τέλος, επισημαίνεται ότι σε αντίθεση με το βοηθητικό πρόγραμμα παραγωγής αντικειμένων, το οποίο έδινε επιπλέον τη δυνατότητα δημιουργίας ενός ορισμένου πλήθους αντικειμένων εφάπαξ, το τρέχον πρόγραμμα επιτρέπει αποκλειστικά τη δημιουργία αλληλεπιδράσεων με αυτόματο επαναλαμβανόμενο τρόπο ανά τυχαία (σύντομα για πρακτικούς λόγους) χρονικά διαστήματα [Εικόνα 4.9]. Αυτό γίνεται καθώς πλησιάζεται περισσότερο η λογική με την οποία προκύπτουν οι αλληλεπιδράσεις σε μία εφαρμογή.



```
Terminal - p17griv@ubuntu: ~/Recommender_Demo/app
File Edit View Terminal Tabs Help
p17griv@ubuntu: ~/Recommender_Demo/app x p17griv@ubuntu: ~/Recommender_Demo/app x
Creating users 5/5
1. Create users
2. Create interactions
Quit: 'q'
> 2
b'303:210'
Publishing interaction: 303 -> 210
b'205:1611'
Publishing interaction: 205 -> 1611
b'205:210'
Publishing interaction: 205 -> 210
b'650:313'
Publishing interaction: 650 -> 313
b'896:1827'
Publishing interaction: 896 -> 1827
```

Εικόνα 4.9: Δοκιμή του Συστήματος - Παραγωγή Αλληλεπιδράσεων

Κεφάλαιο 5

Αξιολόγηση

ΤΟ κεφάλαιο αυτό αφορά το κομμάτι της αξιολόγησης του συστήματος. Πιο συγκεκριμένα, το σύστημα αξιολογείται από μεριάς προγραμματιστή, δηλαδή περιγράφονται τα θετικά του χαρακτηριστικά και οι αδυναμίες του. Στη συνέχεια, αναφέρονται τρόποι με τους οποίους θα μπορούσε να γίνει η αξιολόγηση του συστήματος από τους χρήστες των εφαρμογών.

5.1 Από τη Μεριά του Προγραμματιστή

Το σύστημα, αξιολογείται από μία προγραμματιστική ματιά, αναλύοντας τα πλεονεκτήματα και τις αδυναμίες του τρόπου, με τον οποίο λειτουργεί και επιτυγχάνει το στόχο του, την παραγωγή δηλαδή ποιοτικών συστάσεων κάθε χρονική στιγμή.

Αναφορικά με τα δεδομένα εισόδου, τα οποία λαμβάνει, δίνεται η δυνατότητα στην κάθε εφαρμογή να ορίσει με δικό της τρόπο την έννοια της αλληλεπίδρασης ενός χρήστη με ένα αντικείμενο, εφόσον το σύστημα χρησιμοποιεί δεδομένα έμμεσης ανατροφοδότησης. Για παράδειγμα, σε εφαρμογή παροχής ειδήσεων, αλληλεπίδραση μπορεί θεωρηθεί η ανάγνωση (κλικ) ενός άρθρου από τον χρήστη ή η χρονική διάρκεια κατά την οποία ο χρήστη παραμένει στο άρθρο αυτό (ελέγχεται αν υπερβαίνει ένα συγκεκριμένο κατώφλι). Αντίστοιχα, μία εφαρμογή παρακολούθησης ταινιών ενδεχομένως να βασίζει την έννοια της αλληλεπίδρασης στο χρόνο παρακολούθησης ενός αντικειμένου από τον χρήστη. Επομένως, μπορεί να ειπωθεί ότι το σύστημα προσφέρει μία ευελιξία ως προς αυτό το κομμάτι.

Ένα άλλο βασικό πλεονέκτημα του συστήματος, από προγραμματιστικής άποψης, είναι το γεγονός ότι μπορεί να χρησιμοποιηθεί από μία εφαρμογή, χωρίς να πραγματοποιηθούν ουσιαστικές αλλαγές στην αρχιτεκτονική και τη δομή της. Πιο συγκεκριμένα, το εργαλείο μπορεί να εγκατασταθεί σε διαφορετικό υπολογιστικό σύστημα από αυτό που φιλοξενεί την εφαρμογή, στην περίπτωση μιας εφαρμογής με αρχιτεκτονική τύπου *πελάτη – εξυπηρετητή* (client – server), και να λειτουργεί ανεξάρτητα. Αυτό επιτυγχάνεται καθώς από την μία η εφαρμογή τροφοδοτεί τα δεδομένα αλληλεπιδράσεων μέσω ενός Kafka broker (μπορεί επίσης να εγκατασταθεί και να λειτουργεί ανεξάρτητα) στο σύστημα και από την άλλη η εφαρμογή λαμβάνει κατ' απαίτησή συστάσεις μέσω του εξυπηρετητή συστάσεων του συστήματος.

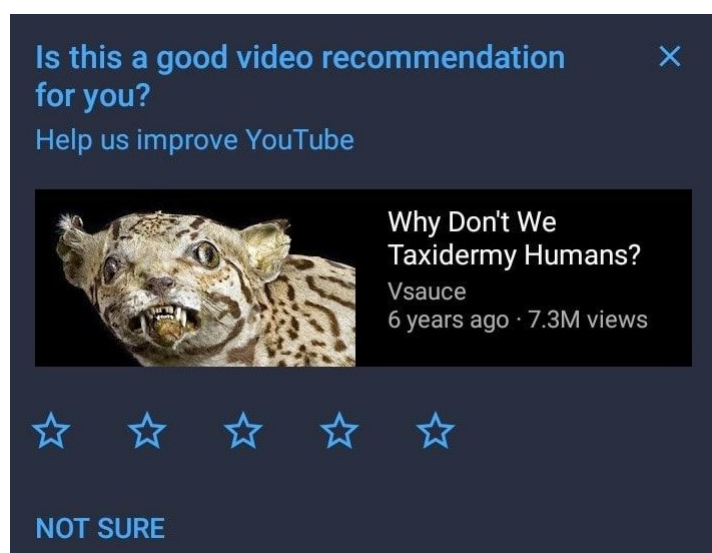
Επίσης, στα θετικά του συστήματος συγκαταλέγεται και η δυνατότητα παραμετροποίησης που παρέχει το σύστημα, ως προς την εκπαίδευση του μοντέλου συστάσεων. Ειδικότερα το σύστημα επιτρέπει να ορίζεται το πλήθος των δεδομένων, τα οποία θα λαμβάνονται σε κάθε εκπαίδευση του μοντέλου αλλά και ο αριθμός των νέων αλληλεπιδράσεων που πρέπει να ληφθούν προκειμένου να γίνει η εκπαίδευση. Έτσι, κάθε εφαρμογή, ορίζοντας αυτές τις παραμέτρους, μπορεί να επιλέγει ανάμεσα στην παροχή συστάσεων υψηλής ακρίβειας, εκπαιδεύοντας το μοντέλο με περισσότερα δεδομένα, αλλά με μία λιγότερο συχνή ενημέρωση του συστήματος (λόγω του μεγαλύτερου φόρτου στους υπολογισμούς), ή στην παραγωγή ενημερωμένων συστάσεων, σύμφωνα με τις πιο πρόσφατες αλληλεπιδράσεις, αλλά με χαμηλότερη ποιότητα. Οι εφαρμογές μπορούν να κάνουν επιλογές τέτοιες ώστε το σύστημα να δουλεύει όπως αυτές επιθυμούν και ταιριάζουν περισσότερο στην πολιτική τους.

Ωστόσο, ο τρόπος με τον οποίο λειτουργεί το σύστημα περιλαμβάνει και ορισμένα σημαντικά μειονεκτήματα. Αρχικά, το ότι χρησιμοποιούνται για την εκπαίδευση του μοντέλου μόνο τα πιο πρόσφατα δεδομένα, μειώνει την ακρίβεια του συστήματος σε ορισμένες περιπτώσεις. Χαρακτηριστικό παράδειγμα είναι η περίπτωση ενός χρήστη, ο οποίος έχει πολύ καιρό να χρησιμοποιήσει μία εφαρμογή και να αλληλεπιδράσει με αντικείμενά της. Είναι πιθανό, τα παλιότερα δεδομένα αλληλεπιδράσεών του να μην χρησιμοποιούνται πλέον στην εκπαίδευση του μοντέλου και άρα οι νέες συστάσεις προς αυτόν τον χρήστη να μην είναι εύστοχες.

Επίσης, είναι προφανές ότι, η συνεχή επανεκπαίδευση του μοντέλου συστάσεων δεν είναι η πιο αποδοτική λύση, καθώς η εκπαίδευση είναι μία διαδικασία ιδιαίτερος απαιτητική σε υπολογιστικούς πόρους, ιδίως όταν ο όγκος των δεδομένων εκπαίδευσης είναι μεγάλος. Παράλληλα, η διαδικασία αυτή απαιτεί έναν συγκεκριμένο χρόνο, ανάλογο με το μέγεθος των δεδομένων εισόδου, για να ολοκληρωθεί. Επομένως, με την κατάλληλη παραμετροποίηση, το μοντέλο συστάσεων μπορεί κάθε φορά να ενημερώνεται σχετικά γρήγορα για τις νέες αλληλεπιδράσεις αλλά δεν είναι δυνατή η ενημέρωσή του σε πραγματικό χρόνο. Αυτό, έχει έναν αντίκτυπο στην ποιότητα των παραγόμενων συστάσεων.

5.2 Από τη Μεριά του Χρήστη

Δεδομένου ότι η αξιολόγηση του συστήματος, ως προς την αποτελεσματικότητά του, βρίσκεται έξω από τα πλαίσια αυτής της εργασίας, περιγράφεται ο τρόπος με τον οποίο θα μπορούσε γίνει. Πιο συγκεκριμένα, μπορούν να υπάρξουν συμπεράσματα για την ποιότητα των παραγόμενων συστάσεων, ζητώντας από τους χρήστες των εφαρμογών την άποψή τους για αυτές [Εικόνα 5.1]. Αυτό μπορεί να επιτευχθεί σχεδιάζοντας κατάλληλες φόρμες, οι οποίες συλλέγουν αυτού του είδους την ανατροφοδότηση από κάθε χρήστη, για τυχαία αντικείμενα που τους είχαν προταθεί στο παρελθόν.



Εικόνα 5.1: Λήψη Ανατροφοδότησης για την Ποιότητα των Συστάσεων – YouTube

Τέλος, πρέπει να σημειωθεί ότι, η αξιολόγηση του συγκεκριμένου συστήματος συστάσεων δεν μπορεί να επιτευχθεί χρησιμοποιώντας τις παραδοσιακές τεχνικές αξιολόγησης τέτοιων συστημάτων. Αυτό συμβαίνει διότι τα δεδομένα εκπαίδευσης του μοντέλου παράγονται δυναμικά. Η τμηματοποίηση των δεδομένων σε σώμα εκπαίδευσης και σώμα ελέγχου και ο υπολογισμός του σφάλματος μεταξύ των προβλέψεων και των πραγματικών τιμών, είναι μία χρονοβόρα διαδικασία, η εφαρμογή της οποίας απομακρύνει το σύστημα από τον στόχο του, δηλαδή την παραγωγή συστάσεων σε σχεδόν πραγματικό χρόνο.

Κεφάλαιο 6

Επίλογος

ΣΤΟ τρέχον και τελευταίο κεφάλαιο αυτής της εργασίας, περιγράφονται αρχικά τα συμπεράσματα, τα οποία προκύπτουν από την εκπόνηση της εργασίας, τη διαδικασία δηλαδή μελέτης, σχεδιασμού και υλοποίησης ενός συστήματος συστάσεων σχεδόν πραγματικού χρόνου. Έπειτα, αναφέρονται ορισμένες προσθήκες – βελτιώσεις για το σύστημα και τον τρόπο λειτουργίας του, οι οποίες μπορούν να πραγματοποιηθούν μελλοντικά.

6.1 Συμπεράσματα

Τα συστήματα συστάσεων είναι εργαλεία λογισμικού, τα οποία αποδεικνύεται ότι είναι πολύ σημαντικά για την επίτευξη των στόχων πολλών εφαρμογών και τη βελτίωση ακόμα περισσότερων. Έως σήμερα, έχουν επινοηθεί πολλαπλές τεχνικές παραγωγής συστάσεων, όπως το συνεργατικό φιλτράρισμα και το φιλτράρισμα με βάση το περιεχόμενο, οι οποίες υποστηρίζονται από διάφορους αλγόριθμους. Κάθε μία από αυτές τις τεχνικές περιλαμβάνει διάφορα προβλήματα, τα οποία οι σχεδιαστές αυτών των συστημάτων καλούνται να αντιμετωπίσουν. Ένα βασικό πρόβλημα, εξ αυτών, που συναντάται στην πλειοψηφία των συστημάτων, είναι το γεγονός ότι αδυνατούν να ενημερώνονται δυναμικά για τις αλλαγές στο περιβάλλον των εφαρμογών, με αντίκτυπο την μείωση της συνολικής τους ακρίβειας.

Συμπεραίνεται, ότι τη λύση σε αυτό το πρόβλημα μπορεί να δώσει ένα σύστημα, το οποίο έχει εξ αρχής σχεδιαστεί για την αντιμετώπισή του. Πιο συγκεκριμένα, προκειμένου οι συστάσεις του συστήματος να είναι ενημερωμένες κάθε χρονική στιγμή, θα πρέπει τα δεδομένα εισόδου να έχουν τη μορφή συνεχούς ροής δεδομένων. Ωστόσο, για την επεξεργασία των δεδομένων αυτής της μορφής ακολουθείται μία διαφορετική προσέγγιση και επιλέγονται διάφορες τεχνικές, όπως το σταθερό ή το κυλιόμενο παράθυρο.

Χρησιμοποιώντας με κατάλληλο τρόπο συνεχούς ροής δεδομένα, τα οποία αντιπροσωπεύουν αλληλεπιδράσεις χρηστών με αντικείμενα, μπορεί να κατασκευαστεί ένα μοντέλο συστάσεων. Για την ενημέρωση αυτού του μοντέλου, προκύπτει ότι μία απλή λύση είναι η επανεκπαίδευση του κάθε φορά που λαμβάνονται νέα δεδομένα. Παρόλα αυτά, η διαδικασία συνεχούς επανεκπαίδευσης είναι μία ιδιαίτερα απαιτητική σε υπολογιστικούς πόρους και χρόνο διαδικασία. Το πρόβλημα αυτό επιδεινώνεται όταν αυξάνεται ο όγκος των δεδομένων εκπαίδευσης και έχει ως αποτέλεσμα την καθυστέρηση της ενημέρωσης του συστήματος. Όμως, ορίζοντας ένα συγκεκριμένο αριθμό των πιο πρόσφατων δεδομένων για την εκπαίδευση του μοντέλου, το σύστημα μπορεί να παραμένει σχετικά ενημερωμένο, να ενημερώνεται δηλαδή σε σχεδόν πραγματικό χρόνο, με αντάλλαγμα μία ενδεχόμενη μείωση στην ποιότητα των συστάσεων.

Τέλος, προκύπτει ότι η αξιοποίηση δεδομένων τύπου έμμεσης ανατροφοδότησης είναι η μόνη επιλογή όταν υπάρχει η απαίτηση το σύστημα συστάσεων να λειτουργεί σε όλες τις εφαρμογές, ανεξάρτητα από το περιεχόμενό τους. Η εκπλήρωση αυτής της απαίτησης, επιτυγχάνεται πιο εύκολα όταν η αρχιτεκτονική του συστήματος είναι τύπου αρχιτεκτονικής μικρο-υπηρεσιών. Πιο συγκεκριμένα, με την υιοθέτηση του μοντέλου ανταλλαγής μηνυμάτων Publish / Subscribe, για την μεταγωγή των δεδομένων εισόδου, και με τη διανομή των συστάσεων μέσω μιας υπηρεσίας τύπου εξυπηρετητή συστάσεων, το σύστημα μπορεί να

λειτουργεί ανεξάρτητα σε οποιαδήποτε εφαρμογή, χωρίς να πραγματοποιηθούν ουσιαστικές αλλαγές στο σύστημα ή στην εφαρμογή.

6.2 Μελλοντικές Προεκτάσεις

Δεδομένου ότι, το σύστημα συστάσεων που σχεδιάστηκε και αναπτύχθηκε στα πλαίσια της εργασίας, έχει κάποιες αδυναμίες και υστερεί σε ορισμένες περιπτώσεις, υπάρχει αρκετός χώρος για ενδεχόμενες μελλοντικές του βελτιώσεις. Αναλυτικότερα, για το λόγο ότι ο τρόπος με τον οποίο επιτυγχάνεται η διαρκής ενημέρωση του συστήματος, με τη συνεχή δηλαδή επανεκπαίδευση του μοντέλου, όπως αναφέρθηκε νωρίτερα, δεν είναι αποδοτικός και η ενημέρωση δεν γίνεται σε πραγματικό χρόνο, είναι ένα κομμάτι, το οποίο χρήζει βελτίωση. Διάφορες μελέτες έχουν γίνει για τη σταδιακή ενημέρωση του μοντέλου συστάσεων, χρησιμοποιώντας τη *Στοχαστική Επικλινή Κάθοδο* (Stochastic Gradient Descent – SGD), έτσι ώστε να μην απαιτείται η επανεκπαίδευσή του [34, 35, 36]. Επομένως, το σύστημα συστάσεων μπορεί να βελτιωθεί εφαρμόζοντας αυτή τη μέθοδο, σε μία μελλοντική του εκδοχή.

Επίσης, περιθώριο βελτίωσης του συστήματος υπάρχει και σε ότι έχει να κάνει με την αντιμετώπιση άλλων προβλημάτων, όπως το πρόβλημα κρύας εκκίνησης, αντιμετώπισης επιθέσεων κακόβουλης τροφοδότησης κ.ο.κ.. Για τα προβλήματα αυτά δεν έχουν ληφθεί αντίμετρα, δεδομένου ότι η αντιμετώπισή τους δεν ανήκει στα πλαίσια αυτής της εργασίας και άρα μπορεί να αποτελέσει μέρος μιας μελλοντικής έρευνας.

Αντίστοιχα, το κομμάτι της αξιολόγησης του υλοποιημένου συστήματος είναι κάτι το οποίο δεν έχει πραγματοποιηθεί στα πλαίσια της εργασίας. Πριν την εφαρμογή οποιασδήποτε βελτίωσης στο σύστημα, θα ήταν χρήσιμη η αξιολόγησή του, προκειμένου να διαπιστωθεί η αποτελεσματικότητά του. Η αξιολόγηση, όπως περιγράφηκε και στο αντίστοιχο κεφάλαιο, μπορεί να γίνει χρησιμοποιώντας το σύστημα σε μία πειραματική ή σε μία πραγματική εφαρμογή και ζητώντας από τους χρήστες ανατροφοδότηση για την ποιότητα των συστάσεων. Έτσι, το κομμάτι της αξιολόγησης του τρέχοντος συστήματος, είναι κάτι το οποίο πρέπει να γίνει στα πλαίσια ενός μελλοντικού πρότζεκτ.

Σε ότι έχει να κάνει με τη διαδικασία για την εγκατάσταση του συστήματος, μπορεί επίσης να γίνει στο μέλλον μία ουσιαστική βελτίωση. Δίνοντας την επιλογή στους κατασκευαστές των εφαρμογών, να εγκαθιστούν το σύστημα συστάσεων με όλα τα επιμέρους προγράμματά του ως ένα ενιαίο πακέτο, η διαδικασία απλουστεύεται σε σημαντικό βαθμό, καθώς μειώνονται τα βήματα και ο χρόνος που απαιτεί. Η δημιουργία ενός τέτοιου πακέτου (container), με όλα τα συστατικά του συστήματος, μπορεί να επιτευχθεί αξιοποιώντας τεχνολογίες εικονικοποίησης σε επίπεδο λειτουργικού συστήματος, όπως είναι το Docker.

Τέλος, πιθανές μελλοντικές προεκτάσεις της μελέτης μπορούν να αποτελούν διάφορες επεκτάσεις του συστήματος αναφορικά με τη λειτουργία του. Πιο συγκεκριμένα, σε μελλοντικές του εκδόσεις, το σύστημα μπορεί να δίνει τη δυνατότητα επιπλέον παραμετροποίησης, για την επιλογή του αλγορίθμου παραγωγής συστάσεων ή ακόμα και διαφορετικής τεχνικής.

Βιβλιογραφικές Αναφορές

- [1] Adomavicius, G., & Tuzhilin, A. (2005). Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions On Knowledge And Data Engineering*, 17(6), 734-749. doi: 10.1109/tkde.2005.99
- [2] Sharma, R., & Singh, R. (2016). Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey. *Indian Journal Of Science And Technology*, 9(20). doi: 10.17485/ijst/2016/v9i20/88005
- [3] Haruna, K., Akmar Ismail, M., Suhendroyono, S., Damiasih, D., Pierewan, A., Chiroma, H., & Herawan, T. (2017). Context-Aware Recommender System: A Review of Recent Developmental Process and Future Research Direction. *Applied Sciences*, 7(12), 1211. doi: 10.3390/app7121211
- [4] Dr. Jain, S., Grover, A., Thakur, P. S., & Choudhary, S. K. (2015). Trends, Problems And Solutions of Recommender System. Kurukshetra, India: National Institute of Technology Kurukshetra.
- [5] Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender Systems: Issues, Challenges, and Research Opportunities. In: Kim K., Joukov N. (eds) *Information Science and Applications (ICISA) 2016. Lecture Notes in Electrical Engineering*, vol 376. Springer, Singapore.
- [6] Oard, D.W., & Kim, J.. (1998). Implicit Feedback for Recommender Systems. College Park, Maryland: Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering.
- [7] Hu, Y., Koren, Y., & Volinsky, C.. (2008). Collaborative Filtering for Implicit Feedback Datasets. Pisa, Italy: Eighth IEEE International Conference on Data Mining. pp. 263-272. doi: 10.1109/ICDM.2008.22
- [8] Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3), 261-273. doi: 10.1016/j.eij.2015.06.005
- [9] Schafer, J.B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative Filtering Recommender Systems. In: Brusilovsky P., Kobsa A., Nejdl W. (eds) *The Adaptive Web. Lecture Notes in Computer Science*, vol 4321. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-540-72079-9_9
- [10] Aberger, C., R. (2014). Recommender: An Analysis of Collaborative Filtering Techniques. Stanford University, California.
- [11] Ajitsaria, A. (2019). Build a Recommendation Engine With Collaborative Filtering – Real Python. Retrieved 2 August 2021, from <https://realpython.com/build-recommendation-engine-collaborative-filtering/>

- [12] Ekstrand, M.D., Riedl, J.T. & Konstan, J.A. (2011). Collaborative Filtering Recommender Systems. University of Minnesota, Minneapolis
- [13] Vinagre, J., Jorge, A.M., & Gama, J. (2014). Evaluation of recommender systems in streaming environments. In: Workshop on Recommender Systems Evaluation: Dimensions and Design (REDD 2014), Held in Conjunction with RecSys. doi: 10.13140/2.1.4381.5367
- [14] Lam S.K., Frankowski D., & Riedl J. (2006). Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. In: Müller G. (eds) Emerging Trends in Information and Communication Security. ETRICS 2006. Lecture Notes in Computer Science, vol 3995. Springer, Berlin, Heidelberg. doi: 10.1007/11766155_2
- [15] Chandramouli, B., Levandoski, J. J., Eldawy, A., & Mokbel, M. F. (2011). StreamRec: A Real-Time Recommender System. Athens, Greece: SIGMOD'11.
- [16] Narkhede, N., Shapira, G., & Palino, T. (2017). Kafka – The Definitive Guide. O'Reilly Media.
- [17] Zeb, M. S. (2020) Comparative Analysis of Data Stream Processing Systems. Jyväskylä, Finland: Faculty of Information Technology, University of Jyväskylä.
- [18] What Is Stream Processing? A Layman's Overview | Hazelcast. (2019). Retrieved 11 August 2021, from <https://hazelcast.com/glossary/stream-processing/>
- [19] Understanding Sliding and Tumbling Windows – Hortonworks Data Platform. (2019). Retrieved 11 August 2021, from https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_storm-component-guide/content/storm-windowing-concepts.html
- [20] Micro-Batch Processing vs Stream Processing | Hazelcast. (2019). Retrieved 12 August 2021, from <https://hazelcast.com/glossary/micro-batch-processing/>
- [21] Tantalaki, N., Souravlas, S., & Roumeliotis, M. (2019). A review on Big Data real-time stream processing and its scheduling techniques. International Journal of Parallel, Emergent and Distributed Systems. doi: 10.1080/17445760.2019.1585848
- [22] Pub/Sub Messaging. Retrieved 16 August 2021, from <https://aws.amazon.com/pub-sub-messaging/>
- [23] Lacic, E., Traub, M., Kowald, D., & Lex, E. (2015). ScaR: Towards a real-time recommender framework following the microservices architecture. In Proceedings of the Workshop on Large Scale Recommender Systems (LSRS2015) at RecSys 2015, 16–20 September 2015, Vienna, Austria (Vol. 15).
- [24] Das, A.S., Datar, M., Garg, A., & Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web (WWW '07). Association for Computing Machinery, New York, NY, USA, 271–280. doi: 10.1145/1242572.1242610
- [25] Fortuna, B., Fortuna, C., & Mladenović, D. (2010). Real-Time News Recommender System. In: Balcázar J.L., Bonchi F., Gionis A., Sebag M. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2010. Lecture Notes in Computer Science, vol 6323. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-15939-8_38

- [26] Adenowo, A.A.A., & Adenowo, B.A. (2013). Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. *International Journal of Scientific & Engineering Research*, (4)7, 427-434
- [27] SDLC – Waterfall Model. (n.d.). Retrieved 23 August 2021, from https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
- [28] Reza, Za.(2015). Matrix Completion via Alternating Least Square (ALS). Scribed by Li, H., He, B., Lublin, M., & Perez, Y. Stanford University, California.
- [29] General Python FAQ — Python 3.8.12 documentation. (n.d.). Retrieved 25 August 2021, from <https://docs.python.org/3.8/faq/general.html>
- [30] Frederickson, B. (2017). Implicit – Fast Python Collaborative Filtering for Implicit Feedback Datasets. Retrieved 25 August 2021, from <https://github.com/benfred/implicit>.
- [31] Thein, K. (2014). Apache Kafka: Next Generation Distributed Messaging System. *International Journal of Scientific Engineering and Technology Research*, vol. 3, no. 47, pp. 9478–9483.
- [32] Apache Kafka. (n.d.). Retrieved 26 August 2021, from <https://kafka.apache.org/>
- [33] Victor. (2017). ALS Implicit Collaborative Filtering. Retrieved 30 August 2021, from <https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe>
- [34] Lin, C., Wang, L., & Tsai, K. (2018). Hybrid Real-Time Matrix Factorization for Implicit Feedback Recommendation Systems. *IEEE Access*, 6, 21369-21380. doi: 10.1109/access.2018.2819428
- [35] Vinagre, J., Jorge, A. M., & Gama, J. (2014). Fast incremental matrix factorization for recommendation with positive-only feedback. In *User Modeling, Adaptation, and Personalization* (pp. 459–470). Springer International Publishing.
- [36] Rendle, S., & Schmidt-Thieme, L. (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. *Proceedings of the 2008 ACM Conference on Recommender Systems – RecSys '08*.
- [37] Grivas, P. (2021). A Near Real-Time Python Recommendation System that can be Integrated in Any Application. <https://github.com/p17griv/real-time-recommendation-service>

Συντμήσεις

ALS	Alternating Least Square
ANN	Artificial Neural Networks
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
IDE	Integrated Development Environment
MAE	Mean Absolute Error
NMAE	Normalized Mean Absolute Error
PIP	Python Package Installer
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SDLC	Software Development Life Cycle
SOM	Self-Organizing Map
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition

Γλωσσάρι Ξενικών Όρων

Aggregations	Συναθροίσεις
Alternating Least Squares	Μέθοδος Εναλλασσόμενων Ελαχίστων Τετραγώνων
Artificial Neural Networks	Τεχνητά Νευρωνικά Δίκτυα
Association Rules	Κανόνες Συσχέτισης
Autoencoders	Αυτοκωδικοποιητές
Cascade Hybrid Systems	Υβριδικά Συστήματα Καταρράκτη
Cluster	Ομάδα
Clustering	Συσταδοποίηση
Cold-Start Problem	Πρόβλημα Κρύας-Εκκίνησης
Collaborative Filtering	Συνεργατικό Φιλτράρισμα
Community-Based Systems	Συστήματα με Βάση την Κοινότητα
Confidence	Εμπιστοσύνη
Context	Πληροφορία των Συνθηκών
Context-Aware	Που Γνωρίζουν για τις Συνθήκες
Cosine Similarity	Ομοιότητα Συνημιτόνου
Coverage	Κάλυψη
Curve Fitting	Ταίριασμα Καμπύλης
Data Science	Επιστήμη των Δεδομένων

Data Stream	Ροή Δεδομένων
Debugging	Αποσφαλμάτωση
Decision Support Accuracy Metrics	Μετρικές Ακρίβειας Υποστήριξης Απόφασης
Demographic Filtering Systems	Συστήματα Δημογραφικού Φιλτραρίσματος
Dimensionality Reduction	Μείωση Διαστάσεων
Event Stream	Ροή Γεγονότων
Explicit Feedback	Δεδομένα Ρητής Ανατροφοδότησης
Exposure	Αποκάλυψη
False Negative	Λανθασμένα Αρνητικά
False Positive	Λανθασμένα Θετικά
Fault Tolerance	Ανθεκτικότητα σε Σφάλματα
Feature Combination Systems	Συστήματα Συνδυασμού Χαρακτηριστικών
Fixed Window	Σταθερό Παράθυρο
F-Measure	F-Μετρική
Framework	Πλαίσιο
Functional Programming	Συναρτησιακός Προγραμματισμός
Hardware	Υλικό
Hybrid Filtering	Υβριδικό Φιλτράρισμα
Hypothesis Systematic Testing	Έλεγχος Υπόθεσης
Implicit Feedback	Δεδομένα Έμμεσης Ανατροφοδότησης
Insights	Γνώσεις
Integrated Development Environment	Ολοκληρωμένο Περιβάλλον Ανάπτυξης

Interpreted	Διερμηνευόμενο
Item-Based	Με βάση το Αντικείμενο
Latent Factors	Κρυφοί Παράγοντες
Learning Rate	Ρυθμός Εκπαίδευσης
Matrix Completion	Συμπλήρωση Πίνακα
Matrix Factorization	Παραγοντοποίηση Πίνακα
Mean Absolute Error	Μέση Απόλυτη Απόκλιση
Memory-Based	Με Βάση τη Μνήμη
Message Processing Guarantee	Εγγύηση Επεξεργασίας Μηνύματος
Meta-Level Hybrid Systems	Υβριδικά Συστήματα Μετα-Επιπέδου
Micro-Batch Processing	Επεξεργασία Μικρο-Ομάδων
Micro-Services	Μικρο-Υπηρεσίες
Missing Data	Ελλείποντα Στοιχεία
Mixed	Μεικτά
Model-Based	Με Βάση το Μοντέλο
Most Popular	Μέθοδος του πιο Δημοφιλούς
Multy-Threading	Πολυ-Νηματικό
Noise	Θόρυβος
Non-personalized	Μη-προσωποποιημένος
Normalized Mean Absolute Error	Κανονικοποιημένη Μέση Απόλυτη Απόκλιση
Novelty	Πρωτοπορία
Nuke Attacks	Επιθέσεις με Σκοπό την Υπονόμευση

Object-Oriented	Αντικειμενοστραφής
Offline Algorithmic Evaluation	Αξιολόγηση Αλγορίθμου σε Δεύτερο Χρόνο
Open-Source	Ανοικτού-Κώδικα
Over-Fitting	Υπερ-Εξειδίκευση
Over-Specialisation	Υπερ-Εξειδίκευση
Packages	Πακέτα
Package Managers	Διαχειριστές Πακέτων
Parallelized Hybrid Systems	Παραλληλοποιημένα Υβριδικά Συστήματα
Partitions	Τμήματα
Personalized	Προσωποποιημένος
Pipelined Hybrid Systems	Διασυνδεδεμένα Υβριδικά Συστήματα
Precision	Ακρίβεια
Precision Recall Curve	Καμπύλη Ακρίβειας Ανάκλησης
Preference	Προτίμηση
Privacy	Ιδιωτικότητα
Probabilistic Model	Πιθανοτικό Μοντέλο
Procedural Programming	Διαδικαστικός Προγραμματισμός
Programming Interface	Προγραμματιστική Διεπαφή
Proxy	Διακομιστής Μεσολάβησης
Publishers	Παραγωγοί
Push Attacks	Επιθέσεις με Σκοποί την Προώθηση
Real-Time	Πραγματικού-Χρόνου

Recall	Ανάκληση
Receiver Operating Characteristic Curve	Καμπύλη Λειτουργικού Χαρακτηριστικού Δείκτη
Recommendation Server	Εξυπηρετητής Συστάσεων
Redundancy	Πλεονασμός
Regression	Γραμμική Παλινδρόμηση
Scalability	Κλιμακωσιμότητα
Self-Organizing Map	Δίκτυα Αυτο-Οργανούμενης Απεικόνισης
Sensitivity	Ευαισθησία
Service-to-Service	Υπηρεσία-προς-Υπηρεσία
Shilling Attacks	Επιθέσεις Κακόβουλης Τροφοδότησης
Singular Value Decomposition	Αποσύνθεση Ιδιαζουσών Τιμών
Sliding Window	Κυλίομενα Παράθυρα
Sparsity	Αραιότητα
Specificity	Εξειδίκευση
State Management	Διαχείριση Κατάστασης
Statistical Accuracy Metrics	Στατιστικές Μετρικές Ακρίβειας
Stochastic Gradient Descent	Στοχαστική Επικλινής Κάθοδος
Streaming Data	Συνεχούς Ροής Δεδομένα
Stream Processing	Επεξεργασία Συνεχούς Ροής Δεδομένων
Stream Processing Engines	Μηχανές Επεξεργασίας Συνεχούς Ροής Δεδομένων
Software Development Life Cycle	Κύκλος Ζωής Ανάπτυξης Λογισμικού
Subscribers	Καταναλωτές
Switching	Εναλλασσόμενα

Testing Set	Σύνολο Δεδομένων Ελέγχου
Topic	Θέμα
Training Set	Σύνολο Δεδομένων Εκπαίδευσης
Transformations	Μετασχηματισμοί
Trend	Τάση
True Positive	Ορθώς Θετικά
Unbounded Dataset	Άνευ Ορίων Σετ Δεδομένων
User-Based	Με Βάση τον Χρήστη
Version Control Systems	Συστήματα Ελέγχου Έκδοσης
Virtualization	Εικονικοποίηση
Waterfall Model	Μοντέλο Καταρράκτη
Web Application	Εφαρμογή Δικτύου
Weighted	Με Βάρη
Windowing	Παράθυρα Επεξεργασίας