Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ένα Μοντέλο Σύστασης για Υπηρεσίες IPTV με Χρόνο-Επίγνωση: Μία Κατά Τμήματα Γραμμική Επέκταση του Πρωτότυπου Αλγορίθμου SVD++** **A Time-Aware Recommendation Model for IPTV: A Partially – Linear Extension to the Original SVD++ Algorithm**
Ονοματεπώνυμο Φοιτητή	**Αντώνιος Σαρρής**
Πατρώνυμο	**Παναγιώτης**
Αριθμός Μητρώο	**ΜΠΣΠ / 12067**
Επιβλέπων	**Γεώργιος Τσιχριντζής, Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2014**

Τριμελής Εξεταστική Επιτροπή

	(υπογραφή)	(υπογραφή)
Γεώργιος Α. Τσιχριντζης Καθηγητής	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

«Το μοναδικό αναφαίρετο δικαίωμά σου είναι να βαδίσεις προς την καταστροφή με όποιον τρόπο εσύ διαλέξεις»

Robert Lee Frost

Ευχαριστίες

Η παρούσα εργασία αποτελεί τη Διατριβή μου στα πλαίσια των μεταπτυχιακών σπουδών στο Τμήμα Πληροφορική του Πανεπιστημίου Πειραιά και πιο συγκεκριμένα στα Πρόγραμμα Μεταπτυχιακών Σπουδών με τίτλο «Προηγμένα Συστήματα Πληροφορικής».

Πρώτα από όλα, θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή Γιώργο Τσιχριντζή που μου εμπιστεύθηκε ένα θέμα με τόσο μεγάλο ενδιαφέρον δίνοντας μου έτσι την ευκαιρία να το μελετήσω σε βάθος.

Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον Δρ. Διονύση Σωτηρόπουλο για την ουσιαστική βοήθεια που μου προσέφερε, παρά τις αυξημένες υποχρεώσεις του, αφιερώνοντας αρκετές ώρες πάνω στο θέμα της διατριβής μου.

Τέλος, θα ήθελα να πω ένα πολύ μεγάλο ευχαριστώ στην Μαρία που με στήριξε και με στηρίζει σε όλες μου τις προσπάθειες.

Περιεχόμενα

Πίνακας Εικόνων / Διαγραμμάτων

Περίληψη

Η παρούσα εργασία αντιμετωπίζει το πρόβλημα της σύστασης τηλεοπτικών προγραμμάτων μέσα στο πλαίσιο της ανάπτυξης υπηρεσιών εξατομίκευσης χρηστών για την IPTV, η επίλυση του οποίου συνεπάγεται την διαχείριση ενός τεράστιου όγκου χρονικών δεδομένων που αφορούν τις επιλογές προγράμματος για ένα ευρύ φάσμα χρηστών και σχετιζόμενων πολυμεσικών αντικειμένων. Συγκεκριμένα, ο πρωταρχικός στόχος αυτής της εργασίας είναι η επέκταση των κλασσικών γραμμικών μοντέλων για την κατασκευή των βασικών εκτιμητών των χρηστών τα οποία είναι ενσωματωμένα στην αρχική διατύπωση του αλγορίθμου σύστασης SVD++. Το θεμελιώδες χαρακτηριστικό αυτών των μοντέλων αφορά στην ομοιόμορφη διαχείριση του άξονα του χρόνου των αλληλεπιδράσεων των χρηστών, αποτυγχάνοντας έτσι να περιγράψουν με ακρίβεια την διακύμανση των αξιολογήσεων των χρηστών με την πάροδο του χρόνου. Εν αντιθέσει, η διατριβή αυτή εισαγάγει ένα περισσότερο ευέλικτο μαθηματικό εργαλείο για την κατασκευή του βασικού εκτιμητή του κάθε χρήστη, χρησιμοποιώντας ένα κατά τμήματα γραμμικό μοντέλο το οποίο κατασκευάζεται υποδιαιρώντας τον χρονικό άξονα των αλληλεπιδράσεων του κάθε χρήστη με τέτοιο τρόπο ώστε σε κάθε υποδιάστημα να αντιστοιχεί ένα προκαθορισμένο μέρος του συνολικού του πλήθους των αξιολογήσεων. Με το τρόπο αυτό, το προτεινόμενο μοντέλο στοχεύει στο να συλλάβει με αποτελεσματικότερο τρόπο σημαντικές μεταβολές στην συμπεριφορά θέασης του χρήστη που διαφορετικά θα αγνοούνταν. Η διενέργεια μια σειράς πειραμάτων επικυρώνουν την αποτελεσματικότητα του εν λόγω μοντέλου όπως αυτή αποτιμάται σε όρους μέσου απόλυτου σφάλματος.

Abstract

This work addresses the problem of recommendation within the context of developing user individualization services for IPTV, involving vast volumes of timely zapping data that span a wide spectrum of users and associated multimedia items. Specifically, the primary objective of this work lies upon the extension of the linear models for the users' baseline predictors which are embedded in the original formulation of the SVD++ recommendation algorithm. The fundamental characteristic of such models concerns the uniform manipulation of the timeline of interactions for each user, thus failing to accurately model the variation of user ratings over time. This work, on the contrary, introduces a more flexible user baseline predictor by utilizing a partially linear model which is built by dividing the timeline of interactions in such a way that a predefined fraction of ratings fall within each time interval. Therefore, the proposed model aims at capturing in a more efficient way significant modifications of user's viewing behavior that otherwise would be ignored. A series of experiments conducted verify the efficiency of the proposed model as measured in terms of Means Absolute Error.

1. Εισαγωγή

Η σημερινή εποχή μεταξύ άλλων χαρακτηρίζεται από την δυνατότητα των ανθρώπων να μεταφέρουν και να ανταλλάξουν πληροφορίες ελεύθερα. Παράλληλα χαρακτηρίζεται και από την επιθυμία αυτών να μπορούν με ευκολία να έχουν άμεση πρόσβαση σε γνώσεις που θα ήταν αδύνατο ή τουλάχιστον δυσκολότερο να βρεθούν στο παρελθόν. Συνεπώς με βάση την παραπάνω έννοια, θα μπορούσε κανείς να χαρακτηρίσει ως έννοια της ψηφιακής εποχής οδηγούμαστε στην αλλαγή από την κλασσική βιομηχανία σε ένα «περιβάλλον» που όχι μόνο χειρίζεται την πληροφορία άλλα βασίζεται σε αυτή.

Το διαδίκτυο αύξησε δραματικά τον όγκο των πληροφοριών που μεταφέρονται. Παράλληλα οι πληροφορίες δεν περιορίζονται μόνο σε κείμενο, αλλά και σε εικόνες, μουσική και βίντεο. Συνεπώς το πρόβλημα της υπερβολικής διάθεσης της πληροφορίας είναι εντονότερο από ποτέ.

Τα συστήματα σύστασης επιχειρούν να αξιοποιήσουν τις ήδη υπάρχουσες σχετικές πληροφορίες και σε συνδυασμό με τη χρήση της επιστήμης της Πληροφορικής να μπορέσουν παρέχοντας λύσεις με τη μορφή εξατομικευμένων υπηρεσιών ή προτάσεων προς όφελος του ανθρώπου.

Στόχος της παρούσας μεταπτυχιακής διατριβής είναι η δημιουργία ενός συστήματος σύστασης τηλεοπτικών προγραμμάτων βασιζόμενο στον αλγόριθμο του Singular Value Decomposition (SVD) με ιδιαίτερη μέριμνα για την παράμετρο του χρόνου. Πρόκειται δηλαδή για επέκταση του κλασσικού μοντέλου που περιγράφεται στη βιβλιογραφία ως αλγόριθμος SVD++.

Συγκεκριμένα στον βασικό εκτιμητή των βαθμολογίων του χρήστη ενσωματώνεται η παράμετρος του χρόνου με τη χρήση ενός κατά τμήματα συνεχούς γραμμικού μοντέλου που υλοποιείται κατακερματίζοντας τον άξονα χρόνου του κάθε χρήστη σε ένα προκαθορισμένο αριθμό σημείων. Με τον τρόπο αυτό το προτεινόμενο μοντέλο στοχεύει στην καλύτερη μοντελοποίηση της μεταβλητής των προτιμήσεων του χρήστη με την πάροδο του χρόνου.

Η παρούσα μεταπτυχιακή εργασία ακολουθεί την παρακάτω δομή:

- **Κεφάλαιο 1**

  Στο κεφάλαιο αυτό παρουσιάζεται η εισαγωγή της παρούσας διατριβής, μια σύντομη περιγραφή του προβλήματος καθώς και του αντικειμένου που πραγματεύεται η εργασία.

- **Κεφάλαιο 2**

  Στο κεφάλαιο δίδεται μία εκτεταμένη βιβλιογραφική διερεύνηση για τη βασική θεωρητική ανάλυση των Συστημάτων Σύστασης [Recommender Systems], με έμφαση στις βασικές παραμέτρους οι οποίες επηρεάζουν τους αλγόριθμους σύστασης.

- **Κεφάλαιο 3**

  Στο κεφάλαιο αυτό παρουσιάζεται η μετάβαση από την κλασσική τηλεόραση στην αμφίδρομη, περιγράφοντας τις επιπρόσθετες σύγχρονες υπηρεσίες της.

- **Κεφάλαιο 4**

  Στο κεφάλαιο αυτό παρουσιάζεται όλη η διαδικασία προ-επεξεργασίας αλλά και διαμόρφωσης των δεδομένων ώστε να καταλήξουν στην τελική μορφή τους για την εισαγωγή στο σύστημα.

- **Κεφάλαιο 5**

  Στο κεφάλαιο αυτό παρουσιάζεται ο Υβριδικός Αλγόριθμος που προτείνεται για το Σύστημα Σύστασης.

- **Κεφάλαιο 6**

  Στο κεφάλαιο αυτό παρουσιάζεται το σύνολο των πειραμάτων αλλά και των αποτελεσμάτων.

- **Κεφάλαιο 7**

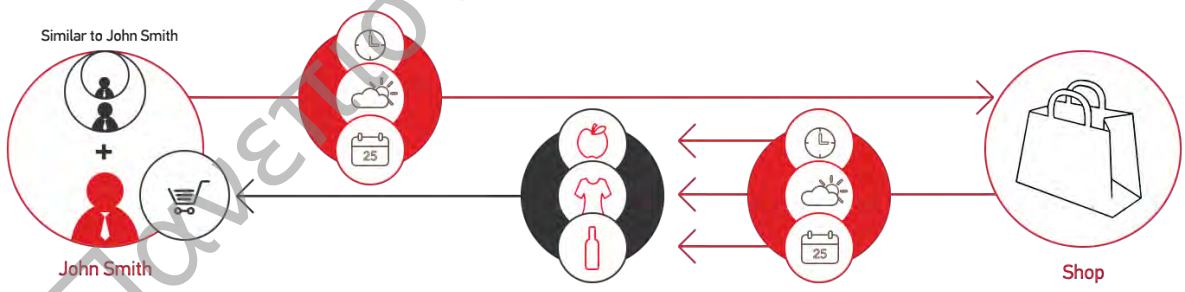
  Στο κεφάλαιο αυτό παρουσιάζεται τα συμπεράσματα και οι μελλοντικές ενέργειες.

2. Μια Θεωρητική Ανάλυση στα Συστήματα Σύστασης

Οι άνθρωποι κατά την διάρκεια της λήψης των καθημερινών τους αποφάσεων, συχνά βασίζονται στις συστάσεις που τους παρέχουν άλλοι. Μακροχρόνιες έρευνες έχουν δείξει ότι βασικό χαρακτηριστικό της ανθρώπινης υπόστασης είναι το ότι ο κάθε άνθρωπος επηρεάζεται από τους άλλους [1]. Για παράδειγμα, ένα άτομα το οποίο θέλει να δει μία ταινία, είναι λογικό να θέλει να βασιστεί στις απόψεις ατόμων που την έχουν ήδη δει ή στην κριτική ενός άρθρου για αυτήν. Επιπλέον, ένας εργοδότης κατά τη διαδικασία πρόσληψης ενδεχομένως να στηρίξει την απόφασή του σε κάποιες συστατικές επιστολές. Σε αυτήν την παρατήρηση βασίστηκε η ανάπτυξη των λεγόμενων Συστημάτων Σύστασης.

2.1. Συστήματα Σύστασης

Συστήματα Σύστασης [Recommender Systems |RS] ή Εισηγητικά Συστήματα [Recommendation systems |RS] (μερικές φορές αντικαθιστώντας το «σύστημα» με ένα συνώνυμο όπως «μηχανή» ή «πλατφόρμα») είναι μια υποκατηγορία των Συστημάτων Φιλτραρίσματος Πληροφοριών [Information Filtering Systems] που επιζητούν να προβλέψουν την «αξιολόγηση» ή «προτίμηση» που θα δώσει ο χρήστης σε ένα στοιχείο [2][3]. Τα Συστήματα Σύστασης έχουν γίνει εξαιρετικά γνωστά τα τελευταία χρόνια, και εφαρμόζονται σε μια ποικιλία εφαρμογών. Οι πιο δημοφιλείς από αυτές είναι ταινίες, μουσική, ειδήσεις, βιβλία, ερευνητικά άρθρα, ερωτήματα αναζήτησης, ετικέτες κοινωνικής δικτύωσης κ.α. Προκειμένου να παραχθούν οι συστάσεις, συλλέγονται από τους χρήστες οι προτιμήσεις τους μέσω μιας βαθμολογίας που δίνουν άμεσα ή έμμεσα για τα αντικείμενα αυτά. Απευθύνονται κυρίως σε άτομα που είτε λόγω της μη επαρκούς προσωπικής εμπειρίας, είτε λόγω της ικανότητας τους, μεγάλος όγκος δεδομένων ή ανεφικτότητα εύρεσης όλης της πληροφορίας, αδυνατούν να αξιολογήσουν τα εν λόγω διαφορετικά αντικείμενα.



Εικόνα 1: Παράδειγμα Συστήματος Σύστασης

Στο σχήμα 1 φαίνεται ένα σύστημα συστάσεων που προτείνει στον χρήστη (John Smith) όχι μόνο να επισκεφτεί ένα κατάστημα για αγορές αλλά και τα προϊόντα που πιθανότατα να ήθελε να αγοράσει. Είναι εμφανές ότι το σύστημα βασίστηκε σε άτομα «φίλια» στον John, χωρίς να είναι απαραίτητο να τα γνωρίζει καθώς και ότι χρησιμοποίησε και επιπλέον πληροφορίες όπως είναι ο χρόνος και οι καιρικές συνθήκες.

2.2. Ανάλυση Μονάδων Συστημάτων Σύστασης

Οι κύριες μονάδες ή μέρη ενός τέτοιου συστήματος είναι η είσοδος, η μέθοδος παραγωγής της σύστασης και η έξοδος. Συγκεκριμένες επιλογές σχεδιασμού τόσο στην είσοδο όσο και στην έξοδο καθορίζουν τις μεθόδους παραγωγής της πρότασης.

2.2.1. Είσοδος RS

Η είσοδος σε ένα σύστημα σύστασης είναι μια συλλογή όπου μεταξύ άλλων περιέχει τους χρήστες με τις προτιμήσεις τους, τα αντικείμενα με τα χαρακτηριστικά τους καθώς και έναν ικανό αριθμό διάφορων συσχετίσεων. Ανάλογα την πηγή που προέρχονται τα δεδομένα διακρίνονται σε εκείνα του χρήστη στον οποίο απευθυνόμαστε ενώ χρειάζεται την τελική σύσταση και σε εκείνα που προέρχονται από τις φίλιες ομάδες του χρήστη.

Η πρώτη κατηγορία δεδομένων χρησιμοποιούνται όπως είναι φανερό ώστε να προκύψουν προτάσεις στοχευμένες αποκλειστικά στον χρήστη, αποφεύγοντας έτσι τις γενικές προτάσεις. Οι κυριότερες κατηγορίες τέτοιων δεδομένων είναι:

- Ρητή Ανατροφοδότηση [Explicit Feedback]: Στην ρητή ανατροφοδότηση τα δεδομένα συγκεντρώνονται για το σύστημα από τους χρήστες με ξεκάθαρο στόχο να το ενημερώσουν για τις επιλογές τους. Παράδειγμα η συμπλήρωση μιας φόρμας ή ενός ερωτηματολογίου.
- Βαθμολογία [Rating]: Θεωρείτε η πιο σαφής μορφή ρητής ανατροφοδότησης. Οι χρήστες βαθμολογούν μόνοι τους τα αντικείμενα που τους ενδιαφέρουν, αναλόγως. Συνήθως χρησιμοποιείται αριθμητική βαθμολογία με κλίμακα 1-5 αλλά είναι αρκετά συχνό σε ειδικές περιπτώσεις να θέλουμε να γνωρίζουμε απλά αν του άρεσε ένα αντικείμενο ή όχι [0|1].
- Σιωπηρή Ανατροφοδότηση [Implicit Feedback]: Στην σιωπηρή ανατροφοδότηση τα δεδομένα συγκεντρώνονται για το σύστημα από την συμπεριφορά της πλοήγησής του, χωρίς ο χρήστης να το αντιλαμβάνεται. Παράδειγμα η εναλλαγή των προγραμμάτων της τηλεόρασης από έναν συγκεκριμένο χρήστη.
- Ιστορικό Αλληλεπίδρασης [Transactions History]: Το ιστορικό ανατροφοδότησης χρησιμοποιείται συνήθως ως μια μορφή σιωπηρής ανατροφοδότησης. Με βάση την ιστορικότητα αλληλεπίδρασης με τα αντικείμενα μπορούν να προκύψουν τα απαραίτητα συμπεράσματα. Παράδειγμα η λίστα με τα προϊόντα που αγοράστηκαν τον τελευταίο χρόνο από έναν συγκεκριμένο πελάτη.
- Λέξεις-Κλειδιά [Keywords]: Επειδή πολλές φορές δεν είναι δυνατό τα δεδομένα να κατανεμηθούν σε μια συγκεκριμένη κατηγορία, μπορούν να χρησιμοποιηθούν λέξεις-κλειδιά. Οι λέξεις-κλειδιά αποδεικνύονται πολύ χρήσιμα για τους σκοπούς τέτοιων συστημάτων, αφού εντοπίζονται επιλογές των χρηστών που είναι ανεξάρτητες από τις κατηγορίες των αντικειμένων.

- Χαρακτηριστικά [Attributes]: Τα χαρακτηριστικά επίσης είναι πολύ χρήσιμα επειδή μπορούν να ομαδοποιήσουν τα δεδομένα μας με πολλούς τρόπους και γίνεται το σύστημά μας πιο ευέλικτο.

Η δεύτερη κατηγορία δεδομένων αποτελείται από μια μεγάλη ποικιλία δεδομένων που προκύπτουν από ολόκληρη την φίλια ομάδα τους χρήστη ή μέρος αυτής. Τέτοια δεδομένα χρειάζονται επεξεργασία κάτω από το πρίσμα των κοινών στοιχείων της κάθε ομάδας.

Κύριες κατηγορίες τέτοιων δεδομένων έχουν αναλυθεί παραπάνω και είναι η ιστορικότητα αλληλεπίδρασης, τα χαρακτηριστικά και η βαθμολόγηση των αντικειμένων, στην περίπτωση ομάδων και όχι μεμονωμένων χρηστών. Επιπλέον μπορούν να προστεθούν οι εξής:

- Δημοφιλία Αντικειμένου [Item Popularity]: Η δημοφιλία ενός αντικειμένου μπορεί να προκύψει είτε από το σύνολο, είτε από μέρος μια ομάδας με συγκεκριμένα χαρακτηριστικά. Παράδειγμα η ακροαματικότητα μία ταινίας.
- Ερμηνευτικά Σχόλια [Comments]: Τα συνολικά σχόλια μια ομάδας ή μέρος αυτής μπορεί να αποδειχθούν πολύ χρήσιμα. Το αρνητικό τους είναι ότι μπορεί να χρειάζονται περισσότερο χρόνο επεξεργασίας, αφού θα πρέπει να ξεχωρίσουν τα θετικά από το αρνητικά σχόλια.
- Υβριδικά Στοιχεία [Hybrid Data]: Ο συνδυασμός των παραπάνω στοιχείων όχι μόνο είναι δυνατός αλλά πολλές φορές επιβάλλεται ώστε να επιφέρει καλύτερα αποτελέσματα.

2.2.2. Μέθοδος Παραγωγής Σύστασης

Τα συστήματα σύστασης χρησιμοποιούν μια σειρά από μεμονωμένους μεθόδους ή συνδυασμό αυτών που χρησιμοποιούνται στο στάδιο της παραγωγής της σύστασης. Οι κυριότερες είναι οι εξής:

- Μη Αυτοματοποιημένη Επιλογή [Manual Selection]: Στη μέθοδο αυτή λαμβάνονται υπόψη δεδομένα που έχουν παραχθεί από τους «ειδικούς» της συγκεκριμένης κατηγορίας ή φίλιας αυτής. Τέτοια δεδομένα εκφράζονται μέσα από κείμενα ή κριτικές. Η μέθοδος αυτή το μόνο που κάνει είναι να αναπαράγει τα στοιχεία αυτά χωρίς να τα επεξεργάζεται. Παράδειγμα όταν ένα χρήστης αναζητεί μια συνταγή θαλασσινών, τότε μπορεί να λάβει ως σύσταση το άρθρο ενός γνωστού σεφ με τις 5 καλύτερες όλων των εποχών.

- Ανάκτηση Ακατέργαστων Στοιχείων [Raw Retrieval]: Στην μέθοδο αυτή υπάρχει η δυνατότητα στον χρήστη να κάνει «ερωτήσεις» στο σύστημα μέσω ενός μηχανισμού αναζήτησης φιλικό προς σε εκείνον. Το σύστημα επεξεργάζεται ανάλογα τα στοιχεία και απαντάει στον χρήστη δίνοντάς του, το βέλτιστο αποτέλεσμα. Παράδειγμα ένας χρήστης αναζητά μια ταινία του ηθοποιού Al Pacino και το σύστημα του βγάζει ως αποτέλεσμα την ταινία που αναζητούσε, δίνοντας του παράλληλα την δυνατότητα να επιλέξει και άλλες ταινίες του συγκεκριμένου ηθοποιού που ενδεχομένως δεν γνώριζε.

- Συσχέτιση Χρήστη προς Χρήστη [User-to-User Correlation]: Στην μέθοδο αυτή τα αντικείμενα που προτείνονται σε ένα χρήστη έχουν συσχέτιση με άλλους χρήστες. Η μέθοδος αυτή είναι γνωστή ως συνεργατική διήθηση [Collaborative Filtering] και είναι προερχόμενη από τεχνικές (Information

Filtering) που για να συστήσουν αντικείμενα στους χρήστες, χρησιμοποιούν προτάσεις ομάδων. Παράδειγμα αν ένας χρήστης ανήκει σε μια ομάδα που έχει δει κατά μεγάλη πλειοψηφία την τελευταία ταινία του Woody Allen και τους άρεσε, ενώ παράλληλα εκείνος δεν την έχει δει, τότε είναι πολύ μεγάλη η πιθανότητα να αρέσει και σε εκείνον.

- Στατιστικές Συνοψίσεις [Statistical Summaries]: Η μέθοδος αυτή περιέχει στατιστικά στοιχεία των προτάσεων μια ομάδας. Τα στατιστικά αυτά μπορεί να είναι ποσοστά μετρήσεων δημοτικότητας κάποιον αντικειμένων ή μέσοι όροι βαθμολογιών. Η μετρήσεις αυτές ενδεχομένως να σχετίζονται τόσο με τους χρήστες όσο και με τα αντικείμενα. Οι χρήστες μπορούν να έχουν γνώση των μέσω τιμών των βαθμολογιών αλλά δεν μπορούν να έχουν πρόσβαση ώστε να βλέπουν ποιος έχει βαθμολογήσει και πόσο. Παράδειγμα ο χρήστης γνωρίζει ότι μια ταινία έχει μέσο όρο βαθμολογίας 6,1 σε έναν σύνολο περίπου 2000 χρηστών.

- Τεχνολογίες βασιζόμενες σε χαρακτηριστικά [Attribute-based Technologies]: Η μέθοδος αυτή όπως προκύπτει και από το όνομά της βασίζεται σε τεχνολογίες με χαρακτηριστικά των αντικειμένων. Μπορεί κανείς σωστά να υποθέσει ότι έχουμε να κάνουμε με μια πιο σύνθετη μορφή της ανάκτησης ακατέργαστων στοιχείων. Παράδειγμα ένας χρήστης μπορεί να βλέπει ταινίες της δεκαετίας του '90 και μόνο αυτές.

- Συσχέτιση Αντικείμενο προς Αντικείμενο [Item-to-Item Correlation]: Στην μέθοδο αυτή σχετίζονται τα πιο συχνά εμφανιζόμενα αντικείμενα, μαζί με τα αντικείμενα που έχει εκφραστεί ενδιαφέρον από τον χρήστη. Σε απλές περιπτώσεις μπορεί να συσχετίσει κατηγορίες που ταιριάζουν για ένα συγκεκριμένο αντικείμενο. Ανάλογα πόσο προηγμένη είναι η μέθοδος μπορεί να κάνει περισσότερους και πιο πολύπλοκους συσχετισμούς. Παράδειγμα στον χρήστη που αγοράζει ένα άσπρο πουκάμισο, προτείνει να αγοράσει και ένα μπλε παντελόνι, πριν τερματίσει την αγορά του, ταιριάζοντας έμμεσα τις κατηγορίες.

Όλες οι μέθοδοι θα έλεγε κανείς ότι επιβάλλεται να επεξεργάζονται τα στοιχεία του χρήστη ή των αντικειμένων, ώστε να μπορεί ο χρήστης να απολαμβάνει τις καλύτερες δυνατές συστάσεις, τόσο στην εγκυρότητα όσο στη αμεσότητα. Κάτι τέτοιο όμως θα ήταν αδύνατο ειδικά στις πιο περίπλοκες μεθόδους όπως User-to-User Correlation & Item-to-Item Correlation όπου απαιτείται μια πιο χρονοβόρα άρα και offline επεξεργασία των δεδομένων που διαθέτουμε. Σε αυτές τις περιπτώσεις μπορεί το μοντέλο μας να χρησιμοποιεί online τα αποτελέσματα που έχουν προκύψει από την offline επεξεργασία και στην συνέχεια να ενημερώνει τα στοιχεία ώστε να γίνεται εκ νέου η offline διαδικασία ώστε να δημιουργηθεί μια αλυσίδα συνεργασίας.

2.2.3. Έξοδος RS

Η έξοδος σε ένα σύστημα σύστασης είναι άρρηκτα συνδεδεμένη με τα χαρακτηριστικά των δεδομένων που αναλύθηκαν στην μέθοδο παραγωγής σύστασης. Το πλήθος, η μορφή αλλά και η ακρίβεια των δεδομένων είναι οι παράμετροι που συντελούν στο αποτέλεσμα. Ο συνηθέστερος τύπος είναι μία καθαρή και συγκεκριμένη σύσταση ή πρόταση. Παράδειγμα συστήνεται στο χρήστη Αντώνη να δει την ταινία «Ο Δικηγόρος του Διαβόλου». Στην περίπτωση που ο χρήστης

ανήκει σε ομάδα γνωστή σε αυτόν π.χ. σε ένα κοινωνικό δίκτυο, μπορεί το σύστημα να κάνει γνωστή την βαθμολογία του. Με αυτό τον τρόπο μπορεί να τον βοηθήσει να εξάγει τα συμπεράσματά του. Επιπλέον, μπορεί να ληφθούν υπόψη τυχόν review άλλων χρηστών της ίδιας ομάδας. Εναλλακτικά μπορεί η έξοδος να είναι της μορφής πρόβλεψης για την ενδεχόμενη βαθμολογία του χρήστη για το αντικείμενο. Συνεπώς το αποτέλεσμα μπορεί να είναι ακόμη και μια λίστα αντικειμένων για έναν χρήστη.

2.3. Εξατομίκευση

Τα σύγχρονα συστήματα σύστασης μπορούν να εξάγουν προτάσεις με διαφορετικό βαθμό εξατομίκευση [Personalization Degree]. Μη Εξατομικευμένα συστήματα [Non Personalization Degree] είναι εκείνα που κάνουν όμοιες προτάσεις σε κάθε χρήστη. Συνήθως βασίζονται σε στοιχεία που έχουν συλλεχθεί από στατιστικά στοιχεία, ερωτηματολόγια και άλλες παρόμοιες τεχνικές. Για παράδειγμα στατιστικά που αφορούν την μεγαλύτερη ακροαματικότητα για ένα πρόγραμμα της τηλεόρασης, επιλογές κριτικών κινηματογράφου ή βιβλίων, σχόλια, μέσοι όροι βαθμολογιών κ.α. Επιπλέον, μερικά συστήματα χρησιμοποιούν την τεχνική της Συνεπής Εξατομίκευσης [Persistent Personalization], δηλαδή συστήματα που εξάγουν προτάσεις που είναι διαφορετικές για διαφορετικούς χρήστες παρόλο που εκείνοι επιλέγουν τα ίδια αντικείμενα. Συστήματα σαν αυτά χρησιμοποιούν συσχέτιση χρήστη προς χρήστη, αντικειμένου προς αντικείμενο αλλά και βασιζόμενα σε χαρακτηριστικά. Επιπρόσθετα, υπάρχουν συστήματα σύστασης που χρησιμοποιούν τα δεδομένα των χρηστών ώστε να προσαρμόσουν τις προτάσεις στα ενδιαφέροντά τους. Αυτά τα συστήματα έχουν ως βάση την συσχέτιση αντικείμενο προς αντικείμενο, μεθόδους που βασίζονται στα χαρακτηριστικά αλλά και συνδυασμό των δυο. Τα προαναφερόμενα συστήματα θεωρείται ότι έχουν Εφήμερη Εξατομίκευση [Ephemeral Personalization].

2.4. Αλγόριθμοι Σύστασης

Οι βασικές μορφές αλγορίθμων σύστασης είναι οι εξής [4]:

- **Συνεργατική Διήθηση [Collaborative Filtering]**

  Τα συστήματα αυτού του τύπου συστήνουν αντικείμενα που είχαν βαθμολογηθεί στο παρελθόν από άλλους χρήστες με όμοια γούστα με τον χρήστη-στόχο, συσχετίζουν λοιπόν τους χρήστες μεταξύ τους. Η ομοιότητα των ενδιαφερόντων δύο χρηστών υπολογίζεται με βάση το πόσο κοινό είναι το παρελθόν των συγκεκριμένων χρηστών στις βαθμολογήσεις.

- **Διήθηση βασισμένη στο περιεχόμενο [Content-based Filtering]**

  Αυτός ο τύπος συστήματος προτείνει αντικείμενα βάσει των αντικειμένων που ο χρήστης είχε βαθμολογήσει σε προηγούμενες αλληλεπιδράσεις του με το σύστημα. Ο υπολογισμός της ομοιότητας (similarity) γίνεται σε σχέση με τα χαρακτηριστικά των αντικειμένων προς σύγκριση. Έτσι για παράδειγμα αν κάποιος είχε δώσει υψηλή βαθμολογία στο παρελθόν σε ένα τραγούδι που ανήκει στην κατηγορία rock, τότε το σύστημα παραγωγής προτάσεων μαθαίνει να συστήνει και άλλα μουσικά κομμάτια που ανήκουν στο συγκεκριμένο είδος.

- **Υβριδικές Προσεγγίσεις [Hybrid Approaches]**

Η κατηγορία αυτή συστημάτων χρησιμοποιεί ένα συνδυασμό των μεθόδων που αναφέρονται παραπάνω, εκμεταλλευόμενα τα προτερήματα της μίας τεχνικής για να καλύψουν τα μειονεκτήματα της άλλης. Υπάρχουν πολλοί διαφορετικοί τρόποι με τους οποίους συνδυάζονται δύο ή και περισσότερες τεχνικές συστημάτων προτάσεων για να δημιουργηθεί ένα υβριδικό σύστημα. Στόχος του συνδυασμού διαφορετικών μεθόδων είναι η βελτίωση της απόδοσής τους.

2.4.1. Συνεργατική Διήθηση [Collaborative Filtering]

Η βασική ιδέα πίσω από τα συστήματα συνεργατικής διήθησης [Collaborative Filtering | CF] είναι η άντληση πληροφοριών από τις εμπειρίες μιας ολόκληρης κοινωνίας χρηστών και όχι μόνο από έναν μεμονωμένο άτομο. Τυπικά κάθε χρήστης σχετίζεται με ένα σύνολο από άλλους χρήστες, των οποίων τα προφίλ παρουσιάζουν τις μεγαλύτερες ομοιότητες (nearest-neighbor users). Οι τεχνικές CF αναζητούν συσχετισμούς ανάμεσα στους χρήστες, οι οποίοι προκύπτουν από τις προτιμήσεις που τους αποδίδονται στα προφίλ των χρηστών. Οι χρήστες που τελικά επιλέγονται είναι εκείνοι που εμφανίζουν το μεγαλύτερο συσχετισμό. Οι χρήστες αυτοί στη συνέχεια αποτελούν "recommendation partners" για το χρήστη που εξετάζουμε και αντικείμενα που εμφανίζονται στα προφίλ τους (και όχι στο προφίλ του χρήστη που εξετάζουμε) μπορούν να προταθούν σε αυτόν.

Επομένως η βαθμολόγηση ενός χρήστη u για ένα αντικείμενο i που δεν το έχει βαθμολογήσει θα είναι όμοια με την βαθμολόγηση που έδωσε στο αντικείμενο ένας άλλος χρήστης του συστήματος, δεδομένου ότι οι δύο χρήστες έχουν βαθμολογήσει άλλα αντικείμενα του συστήματος με παρόμοιο τρόπο. Με το ίδιο σκεπτικό, ο χρήστης u πιθανότατα θα δώσει παρόμοια βαθμολογία για δυο αντικείμενα i και j, αν οι άλλοι χρήστες του συστήματος έχουν δώσει παρόμοιες βαθμολογίες στα δυο αυτά προϊόντα.

Τα βήματα που ακολουθούνται σε ένα Σύστημα CF για την παραγωγή συστάσεων είναι τα εξής:

1. Στάθμιση όλων των χρηστών σε σχέση με την ομοιότητά τους στην συμπεριφορά με τον χρήστη-στόχο
2. Επιλογή μιας ομάδας χρηστών με τα υψηλότερα βάρη που προέκυψαν από το πρώτο βήμα και χρήση τους σαν αναφορά (χρήστες-γείτονες)
3. Πρόβλεψη για την προτίμηση του χρήστη-στόχου για κάποιο αντικείμενο.

Τα Συστήματα Προτάσεων CF μπορούν να κατηγοριοποιηθούν στα memory-based (ή heuristic-based) και στα model-based [8]. Οι memory-based αλγόριθμοι αποτελούν heuristics που κάνουν προβλέψεις βασισμένοι σε μια ολόκληρη συλλογή από προηγούμενα βαθμολογημένα αντικείμενα από τους χρήστες. Η τιμή μιας άγνωστης βαθμολογίας $r_{u,i}$ για το χρήση u και το αντικείμενο i υπολογίζεται σαν η συνάθροιση των βαθμολογιών των N χρηστών-γειτόνων για το ίδιο αντικείμενο i:

$$r_{u,i}=aggr\ r_{u',i}\ ,\ u'\in\hat{U}$$

Όπου $\hat{U}$, είναι το σύνολό των N χρηστών-γειτόνων. Η τιμή του N μπορεί να πάρει τιμές από 1 έως τον αριθμό του πλήθους των χρηστών.

Στην πιο απλή περίπτωση η συνάθροιση είναι ο μέσος όρος:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in \hat{U}} r_{u',i}$$

Ωστόσο η πιο διαδεδομένη προσέγγιση είναι να χρησιμοποιηθεί το σταθμισμένο άθροισμα που φαίνεται παρακάτω:

$$r_{u,i} = k \sum_{u' \in \hat{U}} sim(u, u') \times r_{u',i}$$

όπου ο πολλαπλασιαστής k λειτουργεί σαν παράγοντας κανονικοποίησης και είναι:

$$k = {}^{1}/_{\sum_{u' \in \hat{U}} |sim(u,u')|}$$

Ένα άλλο παράδειγμα συνάθροισης δίνεται από σχέση:

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in \hat{U}} sim(u, u') \times (r_{u',i} - \bar{r}_{u'})$$

Όπου η μέση εκτίμηση του χρήστη u ($\bar{r}_u$), δίνεται από τη σχέση:

$\bar{r}_u = (1/|I_u|) \sum_{i \in I_u} r_{u,i}$ με $I_u = \{i \in I \,|r_{u,i} \neq \emptyset\}$.

Με τον συμβολισμό $r_{u,i} = \emptyset$ δείχνουμε ότι το αντικείμενο i δεν έχει βαθμολογηθεί από τον χρήστη u.

Η συνάρτηση ομοιότητας (similarity) μεταξύ των χρηστών u και u' sim(u, u') είναι μια μετρική απόστασης και χρησιμοποιείται ως ένα βάρος, δηλαδή όσο περισσότερο όμοιοι είναι οι χρήστες u και u' τόσο μεγαλύτερο θα είναι το βάρος της βαθμολόγησης $r_{u',i}$ στην πρόβλεψη του $r_{u,i}$

Οι πιο δημοφιλείς μέθοδοι προσέγγισης είναι η correlation και η cosine-based. Έστω $I_{u,v}$ το σύνολο όλων των αντικειμένων που και οι δυο χρήστες u και v έχουν εκτιμήσει δηλαδή $I_{u,v} = \{i \in I \,|r_{u,i} \neq \emptyset \;\&\; r_{v,i} \neq \emptyset\}$. Στην correlation σύσταση το $I_{u,v}$ είναι συνήθως ένα ενδιάμεσο αποτέλεσμα στη διαδικασία για τον υπολογισμό των «κοντινότερων γειτόνων» του χρήστη u και υπολογίζεται συχνά σαν η τομή των συνόλων I_u και I_v . Όμως υπάρχουν μέθοδοι όπως η graph-theoretic που καθορίσουν τους κοντινότερους γείτονες του u χωρίς τον υπολογισμό του $I_{u,v}$ για όλους τους χρήστες v [9].

Στην περίπτωση αυτή, ο συντελεστής Pearson που χρησιμοποιείται για να μετρήσει την ομοιότητα δίνεται από τη σχέση.

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}$$

Στην περίπτωση της cosine-based οι δύο χρήστες u και v αντιμετωπίζονται σαν δύο διανύσματα στο m-διάστατο χώρο όπου $m = |I_{u,v}|$. Η ομοιότητα μεταξύ των δύο διανυσμάτων μπορεί να μετρηθεί με τον υπολογισμό του συνημίτονου της γωνίας μεταξύ τους:

$$\text{sim}\ (u,v)= \cos(\vec{u},\vec{v}) = \frac{\sum_{i\in I_{uv}} r_{u,i} r_{v,i}}{\sqrt{\sum_{i\in I_{uv}} (r_{u,i})^2}\ \sqrt{\sum_{i\in I_{uv}} (r_{v,i})^2}}$$

Τροποποιήσεις των παραπάνω έχουν προταθεί σε πολλές εργασίες με στόχο τη βελτίωση της απόδοσης. Ενώ οι παραπάνω τεχνικές παραδοσιακά έχουν χρησιμοποιηθεί για να υπολογίσουν τις ομοιότητες μεταξύ των χρηστών, το 2001 οι προτάθηκε να υπολογιστούν οι ομοιότητες μεταξύ των αντικειμένων και να ληφθούν οι εκτιμήσεις από αυτές [10]. Αυτή η ιδέα επεκτάθηκε περαιτέρω στην εργασία των για την παραγωγή των N καλύτερων προτάσεων [11].

Οι model-based μέθοδοι βασίζονται αρχικά σε ένα σύνολο δεδομένων για να εκπαιδεύσουν ένα μοντέλο, το οποίο χρησιμοποιούν στη συνέχεια για να κάνουν τις προβλέψεις. Σύμφωνα με αυτό προτείνεται μια πιθανοτική CF προσέγγιση [8], όπου οι άγνωστες εκτιμήσεις υπολογίζονται ως εξής:

$$r_{u,i}=E(r_{u,i}) = \sum_{j=0}^{n} j \times \Pr(r_{u,i}=j \mid r_{u,i'}, i' \in I_u)$$

Οι τιμές βαθμολόγησης είναι ακέραιοι αριθμοί μεταξύ 0 και n και η έκφραση πιθανότητας είναι η πιθανότητα ότι ο χρήστης u θα κάνει μια ιδιαίτερη εκτίμηση για το αντικείμενο i δεδομένων των προηγουμένων εκτιμήσεων του. Για να υπολογισθεί αυτή η εκτίμηση προτείνονται δύο εναλλακτικά πιθανοτικά μοντέλα: το Cluster model και τα Bayesian δίκτυα. Στην πρώτη περίπτωση «όμοιοι» χρήστες ομαδοποιούνται σε clusters (ομάδες). Λαμβάνοντας υπόψη την ιδιότητα μέλους κάθε ομάδας, οι εκτιμήσεις του χρήστη υποθέτουμε ότι ήταν ανεξάρτητες. Ο αριθμός των ομάδων και οι παράμετροι του μοντέλου εκπαιδεύονται από τα δεδομένα εκπαίδευσης.

Το δεύτερο μοντέλο αντιπροσωπεύει κάθε αντικείμενο στην περιοχή σαν κόμβο σε ένα Bayesian δίκτυο, όπου οι καταστάσεις κάθε κόμβου αντιστοιχούν σε πιθανές εκτιμήσεις για κάθε αντικείμενο. Τόσο η δομή του δικτύου όσο και οι υπό συνθήκη πιθανότητες εκπαιδεύονται από τα δεδομένα. Ένας περιορισμός αυτής της προσέγγισης είναι ότι κάθε χρήστης μπορεί να τοποθετηθεί σε μια μόνο ομάδα.

2.4.2. Διήθηση βασισμένη στο περιεχόμενο [Content-based Filtering | CBF]

Τα Συστήματα που στηρίζονται στη Διήθηση βασισμένη στο περιεχόμενο [Content-based Filtering | CBF] συστήνουν αντικείμενα όμοια με εκείνα που ο χρήστης είχε προτιμήσει στο παρελθόν. Ένα τέτοιο Σύστημα μοντελοποιεί ένα προφίλ για τον χρήστη με βάση τις ιδιότητες των αντικειμένων που έχει παλαιότερα βαθμολογήσει ο ίδιος και στην συνέχεια συνδυάζει τα χαρακτηριστικά που είναι υποθηκευμένα στο προφίλ με τα χαρακτηριστικά του περιεχομένου που αντιστοιχεί στο αντικείμενο, προτείνοντας τελικά ενδιαφέροντα για τον χρήστη αντικείμενα.

Έστω Content(i) είναι το προφίλ ενός αντικειμένου i , δηλαδή ένα σύνολο από χαρακτηριστικά που αντιπροσωπεύουν το περιεχόμενο του i και το χαρακτηρίζουν. Το περιεχόμενο αυτό συνήθως περιγράφεται με λέξεις-κλειδιά (keywords). Η «σημαντικότητα» της λέξης-κλειδί k_j καθορίζεται με μια μετρική

βάρους w_{ij}. Και ContentBasedProfile(u) είναι το προφίλ του χρήστη u που περιλαμβάνει τα χαρακτηριστικά, τα ενδιαφέροντα και τις προτιμήσεις του. Αυτό το προφίλ δημιουργείται από την επεξεργασία του περιεχομένου των αντικειμένων που έχει βαθμολογήσει. Στα συστήματα CBF η συνάρτηση χρησιμότητας ορίζεται ως:

$$f(u,i) = score(ContentBasedProfile(u), Content(i))$$

Τόσο το ContentBasedProfile(u) του χρήστη u όσο και το Content(i) του αντικειμένου i μπορούν να αναπαρασταθούν σαν διανύσματα βαρών $\overrightarrow{w}_u$, $\overrightarrow{w}_i$των λέξεων – κλειδιά. Επιπλέον, συνάρτηση χρησιμότητας βιβλιογραφία συναντάται με κάποια ευρετική βαθμολόγηση που ορίζεται με τους όρους των διανυσμάτων $\overrightarrow{w}_u$, $\overrightarrow{w}_i$, όπως το μέτρο ομοιότητας του συνημίτονου.

$$f(u,i) = \cos(\overrightarrow{w}_u, \overrightarrow{w}_i) = \frac{\sum_{j=1}^{K} w_{j,u}\ w_{j,i}}{\sqrt{\sum_{j=1}^{K} w_{j,u}^2}\sqrt{\sum_{j=1}^{K} w_{j,i}^2}}$$

όπου K είναι ο συνολικός αριθμός των λέξεων-κλειδιά στο σύστημα

Άλλες τεχνικές της κατηγορίας που έχουν επίσης χρησιμοποιηθεί είναι οι Bayesian Classifiers (Ταξινομητές), οι Machine Learning τεχνικές συμπεριλαμβανομένων και τεχνικών Clustering (Συσταδοποίησης), τα Decision Trees (Δέντρων Απόφασης) και τα Artificial Neural Networks (Τεχνητά Νευρωνικά Δίκτυα). Αυτές οι τεχνικές διαφέρουν στο γεγονός ότι υπολογίζουν τις προβλέψεις όχι με ευρυστικό τρόπο π.χ. το μέτρο ομοιότητας συνημίτονου (cosine similarity measure), αλλά βασισμένες σε ένα μοντέλο που εκπαιδεύεται από συλλεγμένα στοιχεία χρησιμοποιώντας στατιστικές και Machine Learning τεχνικές.

Το σημαντικότερο πρόβλημα της μεθόδου CBF είναι η ανάγκη για μια καλή απεικόνιση των γνωρισμάτων περιεχομένου. Η εύρεση αυτής της απεικόνισης μπορεί να αποδειχτεί προβληματική και χρονοβόρα. Σε μερικές περιπτώσεις μάλιστα ίσως δεν είναι δυνατό να βρεθεί μια τέτοια απεικόνιση. Άλλα προβλήματα που εμφανίζουν οι content based τεχνικές έχουν να κάνουν με τον τρόπο που επιλέγουν τα αντικείμενα τα οποία προτείνουν στο χρήστη. Επειδή οι προτάσεις που γίνονται βασίζονται σε αντικείμενα παρόμοια με αντικείμενα που ο χρήστης επέλεξε στο παρελθόν, το προφίλ του χρήστη και τα αντικείμενα που αυτό περιλαμβάνει, περιορίζει αυτόματα το πλήθος των προτάσεων που μπορούν να του γίνουν στο μέλλον. Το πρόβλημα αυτό είναι εντονότερο σε περιπτώσεις νέων χρηστών των οποίων τα προφίλ περιέχουν περιορισμένο αριθμό αντικειμένων. Έτσι είναι πολύ δύσκολο να βρεθούν καινούργια αντικείμενα τα οποία να μοιάζουν με αυτά που υπάρχουν ήδη στο προφίλ τους, για να τους προταθούν.

2.4.3. Υβριδικές Προσεγγίσεις [Hybrid Approaches]

Αρκετά συστήματα συστάσεων χρησιμοποιούν την υβριδική προσέγγιση συνδυάζοντας collaborative και content-based τεχνικές με στόχο να αποφύγουν τους περιορισμούς που ενέχει κάθε μία από αυτές. Οι διαφορετικοί τρόποι που μπορούν να συνδυαστούν οι δύο τεχνικές είναι οι ακόλουθοι:

- Υλοποίηση των τεχνικών ξεχωριστά και συνδυασμός των προβλέψεων.
- Ενσωμάτωση content-based χαρακτηριστικών σε μια collaborative προσέγγιση.
- Ενσωμάτωση collaborative χαρακτηριστικών σε μια content-based προσέγγιση.
- Δημιουργία ενός γενικού μοντέλου το οποίο ενσωματώνει και content-based και collaborative χαρακτηριστικά.

Παρακάτω ακολουθεί Πίνακας που συνοψίζει τις κατηγορίες Συστημάτων Προτάσεων. Στον πίνακα I είναι η ομάδα των αντικειμένων από τα οποία γίνονται οι συστάσεις, U είναι η ομάδα των χρηστών των οποίων οι προτιμήσεις και βαθμολογήσεις είναι γνωστές, u είναι ο χρήστης-στόχος για τον οποίο πρέπει να παραχθεί η σύσταση και i είναι το αντικείμενο για το οποίο ζητείται να προβλεφθεί η βαθμολόγηση του u.

ΤΕΧΝΙΚΗ	**ΔΕΔΟΜΕΝΑ**	**ΕΙΣΟΔΟΣ**	**ΔΙΑΔΙΚΑΣΙΑ**
Collaborative Filtering | CF	Βαθμολογήσεις U στα αντικείμενα του I	Βαθμολογήσεις του u για τα αντικείμενα στο I	Εύρεση χρηστών του U ομοίων του u και χρήση των βαθμολογιών τους για το i
Content-based Filtering | CBF	Χαρακτηριστικά των αντικειμένων στο I	Βαθμολογήσεις του u για τα αντικείμενα στο I	Παραγωγή προφίλ με βάση την βαθμολογική συμπεριφορά του u και χρήση του στο i
Hybrid Approaches	Βαθμολογήσεις U στα αντικείμενα του I και χαρακτηριστικά των προϊόντων στο I	Βαθμολογήσεις του u για τα αντικείμενα στο I	Παραγωγή προφίλ με βάση την βαθμολογική συμπεριφορά του u και χρήση του στο i σε συνδυασμό με Εύρεση χρηστών του U ομοίων του u και χρήση των βαθμολογιών τους για το i

3. Ψηφιακή Τηλεόραση

Η λέξη τηλεόραση προέρχεται από το αρχαίο ελληνικό πρόθεμα «τηλε» που σημαίνει μακριά και της λέξης «όραση» που σημαίνει βλέπω. Η τηλεόραση είναι ένα σύστημα τηλεπικοινωνίας που χρησιμεύει στη μετάδοση και λήψη κινούμενων εικόνων και ήχου εξ αποστάσεως. Αποτελεί το κυριότερο και δημοφιλέστερο Μέσο Μαζικής Επικοινωνίας και η χρήση της είναι ιδιαίτερα διαδεδομένη σε όλο τον κόσμο.[7] Ο όρος καλύπτει ολόκληρο το φάσμα των τεχνικών χαρακτηριστικών και των δραστηριοτήτων που αφορούν τα τηλεοπτικά προγράμματα, καθώς και τη μετάδοσή τους. Συνήθως, λέγοντας "τηλεόραση" εννοούμε τη συσκευή, δηλαδή τον δέκτη, ο οποίος λαμβάνει το (τηλεοπτικό) σήμα που εκπέμπουν οι τηλεοπτικοί σταθμοί σε συγκεκριμένες συχνότητες (ή αλλιώς κανάλια) με την οθόνη που απεικονίζει το αποτέλεσμα της εκπομπής (μετατροπή του σήματος σε εικόνα και ήχο).

3.1. Η εξέλιξη της Τηλεόρασης

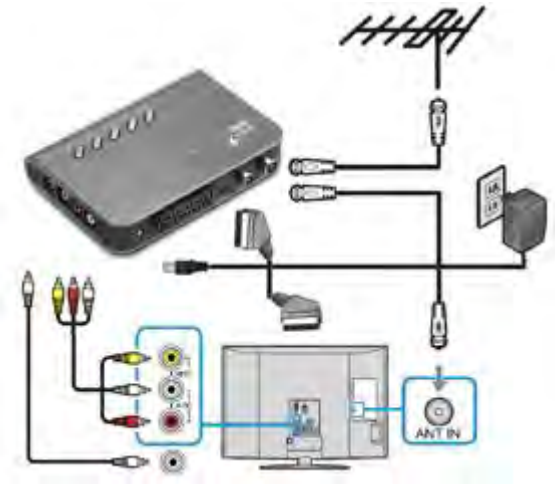
Οι πρώτες τηλεοπτικές μεταδόσεις με επιτυχία έγιναν μεταξύ 1928 – 1935 στο Λονδίνο, από τον John Logie Baird. Στο αρχικό στάδιο του συστήματος στις εικόνες δεν μπορούσαν να αναπαραχθούν οι μικρές λεπτομέρειες καθώς τις αποτελούσαν μόλις 30 γραμμές. Στις 25 Ιουνίου του 1951 μεταδίδεται η πρώτη έγχρωμη εκπομπή από το αμερικανικό κανάλι CBS. Η αύξηση της δημοτικότητας της τηλεόρασης συνεχίστηκε φτάνοντας στα τέλη της δεκαετίας του 1980 να λειτουργούν στην Αμερική περίπου 1300 τηλεοπτικοί σταθμοί, ενώ το 98% των κατοικιών να διαθέτει έστω μία τηλεόραση. Στην δεκαετία του 2000, οι τηλεοράσεις άρχισαν να έχουν ιδιαίτερα όμορφη σχεδίαση. Πιο συγκεκριμένα, να κατασκευάζονται λεπτές, με μεγάλο μέγεθος απεικόνισης και ανάλυσης. Οι περισσότερες τηλεοράσεις συνδέονται στο διαδίκτυο, συνδέονται με ιδιαίτερα εξελιγμένες παιχνιδομηχανές, φορητές συσκευές ή computers προσφέροντας πολλές ώρες διασκέδασης στους χρήστες. Παράλληλα γίνεται η σταδιακή μεταφορά από την αναλογική στην ψηφιακή τηλεόραση, επιτρέποντας της αύξηση της ποιότητας μετάδοσης βίντεο και ήχου.

3.2. Digital Video Broadcasting [DVB]

Διεθνώς η πιο δημοφιλής σουίτα με ανοικτά πρότυπα για την ψηφιακή τηλεόραση είναι το Digital Video Broadcasting [DVB]. Με βάση την μέθοδο μεταφοράς σήματος μπορούμε να προβούμε στην εξής κατηγοριοποίηση:

- Επίγεια Ψηφιακή Μετάδοση [DVB-T]
- Δορυφορική Τηλεόραση [DVB-S | DVB-S2]
- Καλωδιακή Τηλεόραση [DVB-C]
- Επίγεια Ψηφιακή Μετάδοση για φορητές συσκευές [DVB-H]

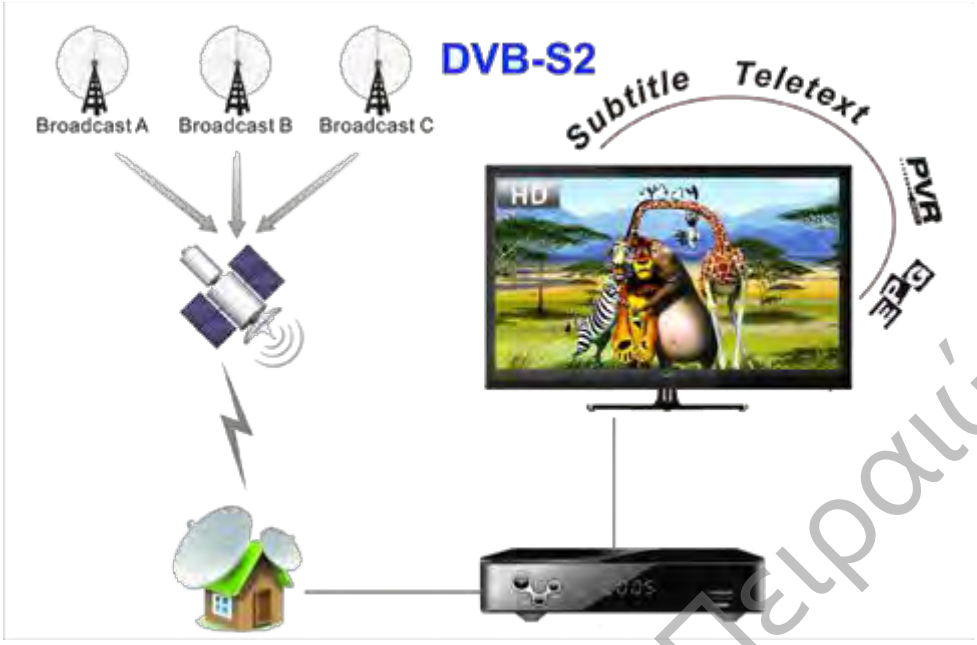
3.2.1. Επίγεια Ψηφιακή Μετάδοση [DVB-T]



Εικόνα 2: Συνδεσμολογία DVB-T

Το Digital Video Broadcasting – Terrestrial [DVB-T] είναι το Ευρωπαϊκό σύστημα επίγειας εκπομπής και λήψης ψηφιακού σήματος. Με βάση αυτό το σύστημα εκπέμπεται μια ροή συμπιεσμένης ψηφιακής εικόνας και ήχου συμπιεσμένη κατά MPEG-2 ή H.264 και διαμόρφωση OFDM. Όσο για την λήψη γίνεται με απλή κεραία. Αφού πραγματοποιηθεί η κωδικοποίηση της εικόνας, του ήχου και των δεδομένων πολυπλέκονται σε μια ροή προγράμματος MPEG-2 PS, εκτός αν στο ίδιο κανάλι μεταδίδονται και άλλα προγράμματα οπότε έχουμε MPEG-2 TS. Η λήψη γίνεται μέσα από τον δέκτη γνωστό και ως Set Top Box. Οι ρυθμοί μετάδοσης κυμαίνονται από 5-32 Mbit/s ανάλογα με την διαμόρφωση και την κωδικοποίηση.

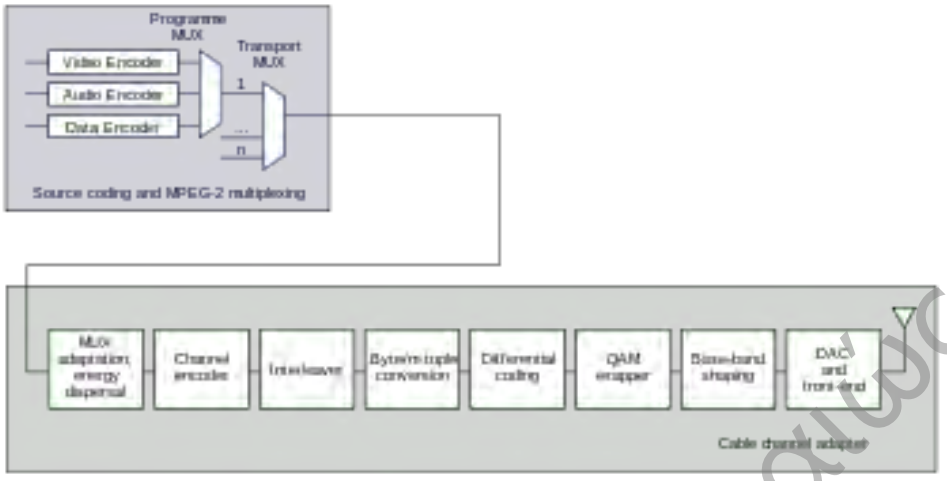
3.2.2. Δορυφορική Τηλεόραση [DVB-S | DVB-S2]



Εικόνα 3: Συνδεσμολογία DVB-S2

Από το 1995 χρονολογείται το DVB-S [Digital Video Broadcasting Satellite] και αποτελεί το πρωταρχικό σύστημα ψηφιακής μετάδοσης βίντεο μέσω δορυφόρων. Αρχικά τα βίντεο προς μετάδοση ήταν κωδικοποιημένα ως MPEG-2 έχοντας ποιότητα ανάλυσης SDTV [Standard Definition TV]. Το 2005 το βελτιωμένο DVB-S2 [Digital Video Broadcasting Satellite - Second Generation] έχοντας ποιότητα ανάλυσης HDTV [High Definition TV], αντικατέστησε το υπάρχον DVB-S. Τα βίντεο είναι κωδικοποιημένα με τον αλγόριθμο H.264 [MPEG-4] με δυνατότητα εκπομπής με κωδικοποίηση MPEG-2 και διαμόρφωση QPSK ή MAPSK. Τα δεδομένα πριν την εκπομπή κατανέμονται σε πακέτα DVB-S|2 εφαρμόζοντας κωδικοποίηση CRC-8 για την διόρθωση των λαθών. Οι τρόποι διαμόρφωσης είναι QPSK, 8 PSK, 16 APSK και 32 APSK. Χάρις τις νέες τεχνικές, όπως η αλλαγή παραμέτρων διαμόρφωσης και κωδικοποίησης [VCM - Variable Coding and Modulation] σε πραγματικό χρόνο και την συμπίεση MPEG-4 επιτυγχάνει 30% καλύτερη απόδοση από το DVB-S.

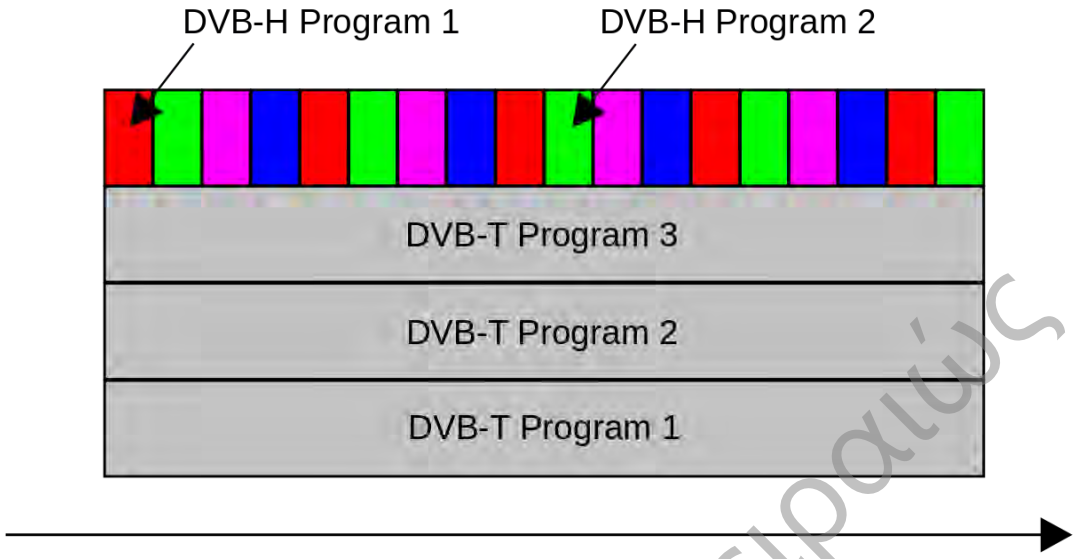
3.2.3. Καλωδιακή Τηλεόραση [DVB-C]



Εικόνα 4: Ανταλλασσόμενη Πληροφορία Πακέτων σε DVB-C

Το σύστημα ψηφιακού βίντεο καλωδιακής μετάδοσης DVB-C [Digital Video Broadcasting – Cable] επιτυγχάνει να μεταδίδει βίντεο και ήχο συμπιεσμένα με τον αλγόριθμο MPEG-2 και υπό QAM διαμόρφωση. Οι ροές προγράμματος πολυπλέκονται σε μια μορφή μεταφοράς MPEG-2 TS [Transport Stream]. Ανάλογα από τις παραμέτρους διαμόρφωσης, οι ρυθμοί μετάδοσης του MPEG-2 TS μπορούν να φτάσουν τα 64 Mbit/s. Στα πακέτα δεδομένων μήκους 188 bytes εφαρμόζεται ο κώδικας RS [Reed Solomon], με δυνατότητα διόρθωσης ως 8 λάθος bytes. Οι τρόποι διαμόρφωσης είναι 16 QAM, 32 QAM, 64 QAM, 128 QAM και 256 QAM. Παράλληλα το διαμορφωμένο σήμα φιλτράρεται λόγω της απομάκρυνσης των παρεμβολών και στην συνέχεια μετατρέπεται σε αναλογικό σήμα για να διαμορφωθεί στο τελικό σήμα RF.

3.2.4. Επίγεια Ψηφιακή Μετάδοση για φορητές συσκευές [DVB-H]



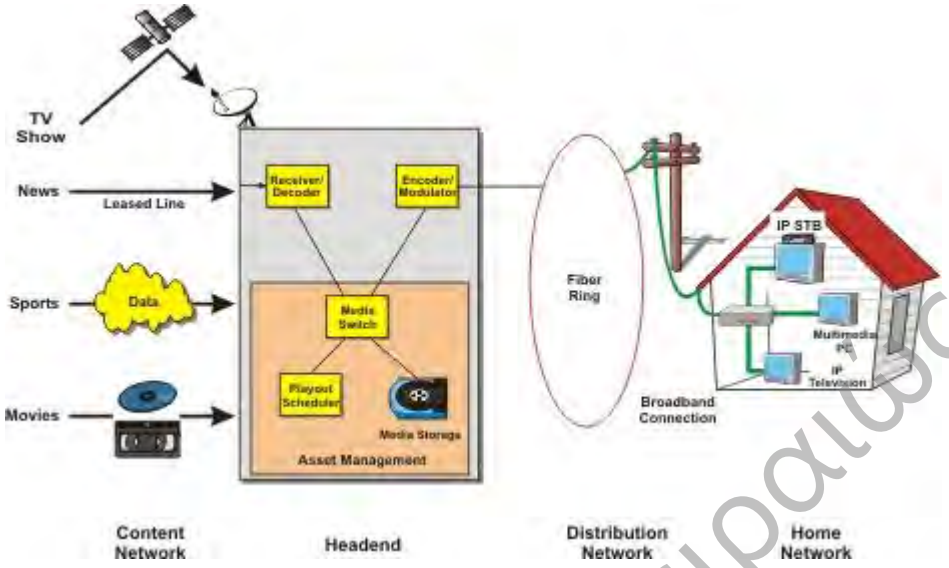
Εικόνα 5: Απεικόνιση DVB-H Βάσει Χρόνου

Προς τα τέλη του 2004, με σκοπό την ψηφιακή μετάδοση βίντεο σε φορητές συσκευές, δημιουργήθηκε το DVB-H [Digital Video Broadcasting – Handheld] καθιστώντας την πιο νέα τεχνική μετάδοσης του DVB. Η τροποποίηση αφορούσε ένα μονόδρομο κανάλι υψηλής ταχύτητας ενώ παράλληλα χρησιμοποιήθηκε η τεχνική «time slicing» για την οικονομία της ισχύος. Πιο συγκεκριμένα το DVB-H λειτουργεί στις μπάντες VHF-III [174-230 MHz], UHF-IV/UHF-V [470-830 MHz] και L [1.452-1.492 GHz]. Κάθε κανάλι εκπομπής εκπέμπει πολλά προγράμματα σε μια καθορισμένη σειρά, μόνο όμως για συγκεκριμένα χρονικά διαστήματα κάνοντας οικονομία στην ενέργεια, αφού απενεργοποιούνται προγράμματα που δεν μας ενδιαφέρουν.

3.3. Αμφίδρομη | Διαδραστική Τηλεόραση

Ωστόσο, το σύστημα DVB εξακολουθεί να είναι μια μονόδρομη τηλεόραση. Αυτό σημαίνει ότι δεν διαθέτει διαδραστικές λειτουργίες που να επιτρέπουν στους χρήστες να επηρεάσουν την αναπαραγωγή. Η επόμενη γενιά τηλεόρασης επιβάλλεται να είναι αμφίδρομη με ορισμένα διαδραστικά χαρακτηριστικά. Παράδειγμα ο χρήστης μπορεί να διακόψει προσωρινά την ζωντανή μετάδοση και να την συνεχίσει σε επόμενο στιγμή. Επίσης υπάρχει η δυνατότητα να δει ένα πρόγραμμα 24 ώρες μετά την ζωντανή μετάδοση αυτού.

3.4. Internet Protocol Television [IPTV]



Εικόνα 6: Βασική Συνδεσμολογία IPTV

Το Internet Protocol Television (IPTV) [5|6] είναι ένα σύστημα που δίνει την δυνατότητα να μεταδίδεται η υπηρεσία της ψηφιακής τηλεόρασης με την βοήθεια του πρωτοκόλλου του διαδικτύου, μέσω μιας ευρυζωνικής σύνδεσης. Πιο απλοποιημένα το τηλεοπτικό σήμα αντί να παραλαμβάνεται από τον χρήστη μέσω των παραδοσιακών τρόπων που έχουν περιγραφεί και παραπάνω, παραλαμβάνεται από τον χρήστη μέσω τεχνολογιών των δικτύων υπολογιστών. Η αποκωδικοποίηση του σήματος γίνεται μέσω ενός Set Top Box που στην συνέχεια στέλνει το σήμα στην οθόνη για την μετάδοση του. Το πλεονέκτημα αυτού του συστήματος είναι ότι οι χρήστες έχουν την δυνατότητα να εμφανίσουν το περιεχόμενο σε οποιαδήποτε οθόνη ακόμη και αν είναι φορητής συσκευής. Οι providers έχουν την δυνατότητα, στεγάζοντας Servers, να αποθηκεύουν πολυάριθμους τίτλους ταινιών, προσφέροντας στους χρήστες το περιεχόμενό τους. Το περιεχόμενο αυτό μπορεί να γίνει μετάδοση ή διανομή κατά απαίτηση του χρήστη. Όλα τα αιτήματα των χρηστών φτάνουν στον server, ώστε να προχωρήσει στην μετάδοση των συγκεκριμένων ροών, αφού τεμαχιστούν σε IP πακέτα δεδομένων όπου και απαιτεί το πρωτόκολλο του δικτύου. Τα συμπιεσμένα πακέτα video δεδομένων απαιτούν για τις ανάγκες των χρηστών τουλάχιστον 1 Mbit/s. Συνεπώς για την ομαλή ροή, ο provider θα πρέπει να εξασφαλίσει συνδέσεις «αφιερωμένες», εξασφαλίζοντας την αδιάκοπη λειτουργία.

3.4.1. Υπηρεσίες IPTV

Οι υπηρεσίες IPTV είναι αρκετά πιο εξελιγμένες από την κλασσική ψηφιακή τηλεόραση, τόσο στο διαδραστικό μέρος όσο και στις «On Demand» υπηρεσίες.

Υπάρχουν με βάση τις υπηρεσίες οι εξής δυο μεγάλες κατηγορίες, καθώς και ο συνδυασμός τους:

- **Υπηρεσίες Broadcast:** Οι υπηρεσίες που προσφέρουν την ταυτόχρονη μετάδοση του περιεχομένου με μεγάλο πλεονέκτημα την ιδιαίτερα μικρή επιβάρυνση στο δίκτυο. Παράδειγμα το Linear πρόγραμμα των καναλιών που υπάρχουν και στην ψηφιακή τηλεόραση.
- **Υπηρεσίες «On Demand»:** Οι υπηρεσίες που βασίζονται στην διαφορετική μετάδοση του περιεχομένου ανά χρήστη. Ο χρήστης είναι εκείνος που δηλώνει το ενδιαφέρον για το περιεχόμενο. Παράδειγμα η επιλογή του χρήστη να δει μια συγκεκριμένη ταινία μια δεδομένη στιγμή που θα την επιλέξει εκείνος.
- **Υβριδικές Υπηρεσίες:** Υπάρχουν και υπηρεσίες που συνδυάζουν και τους δυο παραπάνω τρόπους. Με αυτόν τον τρόπο μπορεί ο χρήστης να απολαμβάνει τα πλεονεκτήματα και των δυο κατηγοριών, ανάλογα με τις ανάγκες του.

3.4.2. Υπηρεσίες Broadcast

Το broadcasting αφορά κυρίως την κλασσική ψηφιακή τηλεόραση [Linear Program]. Στοχεύει κυρίως σε χρήστες που αρέσκονται να τους παρέχουν ή αν θέλετε να τους «σερβίρουν» το πρόγραμμα άλλοι αντί για τους ίδιους. Τέτοιοι χρήστες είτε δεν έχουν τον χρόνο είτε δεν θέλουν να τον ξοδέψουν για αυτό τον σκοπό, το μόνο που θέλουν είναι να ανοίξουν την τηλεόραση και να βρουν γρήγορα αυτό που τους ταιριάζει ώστε να διασκεδάσουν. Έτσι ο Provider οργανώνει και ταξινομεί ένα μεγάλο αριθμό καναλιών, παρουσιάζοντας στον χρήστη το περιεχόμενό τους. Ο χρήστης έχει την δυνατότητα να επιλέξει τι θα παρακολουθήσει, ανάλογα την θεματική του κάθε προγράμματος ή καναλιού, έχοντας την δυνατότητα να ταξινομήσει και ο ίδιος τα κανάλια που του προσφέρονται.

Επιπρόσθετα έχει την δυνατότητα σε μια σειρά από υπηρεσίες όπως:

- **Parental Control:** Δυνατότητα του χρήστη όπου του επιτρέπει να κλειδώσει | ξεκλειδώσει κάποια κανάλια μόνιμα ή προσωρινά, ώστε να προστατέψει π.χ. τα παιδιά του από το ακατάλληλο πρόγραμμα.
- **Multicast [Live Radio]:** Σύνολο υπηρεσιών ήχου όπου ο χρήστης έχει την δυνατότητα να ακούσει μουσική μέσω της τηλεόρασης. Αυτό μπορεί να γίνει με την σύνδεση κάποιου ζωντανού προγράμματος ραδιοφωνικού σταθμού.
- **Pay Per View [PPV]:** Προσφέρει την δυνατότητα στους χρήστες να παρακολουθήσουν ένα συγκεκριμένο πρόγραμμα χωρίς να είναι συνδρομητές στο συγκεκριμένο κανάλι που εκπέμπεται. Για παράδειγμα ο χρήστης θέλει να παρακολουθήσει έναν σημαντικό ποδοσφαιρικό αγώνα, χωρίς να πληρώσει συνδρομή ενός έτους.
- **Near Video on Demand [NVoD]:** Στην πραγματικότητα είναι κανάλια που δημιουργούνται και προγραμματίζονται από τον provider. Σε κάθε κανάλι από αυτά γίνεται broadcast το περιεχόμενο αλλά με την διαφορά ότι είναι μετατοπισμένο σε σταθερά χρονικά διαστήματα [staggering time]. Ο χρήστης έχει την δυνατότητα να κάνει Fast Forward ή Rewind αλλά αυτό που συμβαίνει στην πραγματικότητα είναι να αλλάζει κανάλι και να βλέπει κάποιο που παίζει το περιεχόμενο μετατοπισμένο κατά λίγο χρονικά.

3.4.3. Υπηρεσίες «On Demand»

Οι «On Demand» υπηρεσίες απευθύνονται σε πιο «δύσκολους» χρήστες που δεν τους αρκεί αυτό που παίζεται ζωντανά εκείνη την στιγμή στο αρχικό πρόγραμμα των καναλιών. Οι χρήστες αυτοί είτε δεν έχουν τον χρόνο να είναι μπροστά στην τηλεόραση την συγκεκριμένη που παίζεται το αγαπημένο τους πρόγραμμα, είτε δεν θέλουν να μπουν στην διαδικασία να το προγραμματίσουν, είτε απλά δεν τους καλύπτει αυτό που παίζει εκείνη την στιγμή. Για τους χρήστες αυτούς ο provider έχει δημιουργήσει μια μεγάλη συλλογή με πλούσιο περιεχόμενο έτοιμο να ικανοποιήσει και τον πιο απαιτητικό χρήστη, την στιγμή που θα το επιλέξει. Πρόκειται για μια καθαρά unicast τεχνολογία.

Επιπρόσθετα έχει την δυνατότητα σε μια σειρά από υπηρεσίες όπως:

- **Unicast [Music On Demand]:** Σύνολο υπηρεσιών ήχου όπου ο χρήστης έχει την δυνατότητα να ακούσει μουσική μέσω της τηλεόρασης. Αυτό μπορεί να γίνει με την πρόσβαση του χρήστη σε μια μεγάλη συλλογή από τραγούδια ώστε να κάνει την επιλογή του.
- **Video on Demand [VoD]:** Η συγκεκριμένη υπηρεσία επιτρέπει σε έναν χρήστη να επιλέξει την παρακολούθηση video-περιεχόμενου μέσα από μια συλλογή που έχει δημιουργήσει ο provider. Επιπλέον έχει στην διάθεσή του μια πληθώρα από ευκολίες & λειτουργίες όπως Pause, Fast Forward, Rewind. Το περιεχόμενο αυτό είναι διαθέσιμο στον χρήστη για περιορισμένο χρονικό διάστημα και μπορεί να είναι δωρεάν ή να χρεώνεται ανά μονάδα. Τεχνικά όταν ο χρήστης επιλέξει το video-περιεχόμενο, αυτό διανέμεται μέσω ξεχωριστού Unicast Stream το οποίο δημιουργείται από το VoD server με κατάληξη το Set Top Box.
- **Subscription VoD [SVoD]:** Πρόκειται για μία νέα υπηρεσία. Προσφέρει απεριόριστη χρήση του video-περιεχομένου και χρέωση με πάγιο. Η ποικιλία και το πλήθος των επιλογών είναι τέτοιο που απέκτησε σύντομα πολλούς υποστηρικτές.

3.4.4. Υβριδικές Υπηρεσίες

Οι υβριδικές υπηρεσίες έχουν τον πλεονέκτημα ότι δεν απευθύνονται σε ειδικές κατηγορίες χρηστών αλλά σε όλους ανεξαρτήτως. Αυτό συμβαίνει διότι κάθε χρήστης μπορεί να χρησιμοποιήσει την υπηρεσία με γνώμονα την προσωπική του χρήση χωρίς την παρέμβαση του provider.

Επιπρόσθετα έχει την δυνατότητα σε μια σειρά από υπηρεσίες όπως:

- **Time Shifted TV [TSTV]:** Δίνει τη δυνατότητα για τις λειτουργίες Pause, Fast Forward & Rewind στο ζωντανό πρόγραμμα της τηλεόρασης. Τεχνικά το πρόγραμμα που προβάλλεται εκείνη την στιγμή αποθηκεύεται είτε στο STB, είτε στον Server του provider (για να αποσταλεί αργότερα μέσω unicast stream), την στιγμή που ο χρήστης πατήσει Pause ή Reward. Δηλαδή συνδυάζει Broadcasting & On Demand.

- **Catchup:** Ως επέκταση του Time Shifted, δίνει την δυνατότητα της παρακολούθησης ενός προγράμματος σε μεταγενέστερο χρόνο, όταν το επιλέξει ο χρήστης. Δηλαδή στην ουσία ένα ζωντανό πρόγραμμα μπορεί να μετατραπεί σε VoD. Τεχνικά ο provider καταγράφει σε έναν server όλο το

ζωντανό πρόγραμμα ενός πλήθους καναλιών και το προσφέρει για κάποιο περιορισμένο χρονικό διάστημα στον χρήστη.

- **Personal Video Recorder [PVR]:** Η υπηρεσία αυτή επιτρέπει στον χρήστη να εγγράψει το πρόγραμμα που παρακολουθεί και να το παρακολουθήσει ετεροχρονισμένα. Αυτό μπορεί να γίνει είτε στο STB, είτε στον server του provider. Η παραπάνω διαδικασία μπορεί να πραγματοποιηθεί είτε άμεσα με το πάτημα ενός πλήκτρου εγγραφής, είτε με αυτόματο τρόπο και σε προγραμματισμό ημερήσιας, εβδομαδιαίας ή μηνιαίας βάσης. Δεν ξενίζει τον χρήστη καθόλου η εμπειρία μιας τέτοιας υπηρεσίας αφού ο τρόπος εγγραφής και στο παρελθόν γινότανε με τον ίδιο τρόπο (VCR). Παράλληλα αξίζει να αναφερθεί ότι το πρόγραμμα που γράφτηκε από τον χρήστη, δεν είναι διαθέσιμο μόνο για κάποιο περιορισμένο διάστημα, αλλά για όσο διάστημα επιθυμεί ο ίδιος, αφού ο μόνος περιορισμός είναι ο αποθηκευτικός χώρος.

4. Προ-επεξεργασία | Διαμόρφωση Δεδομένων

Το στάδιο της προ-επεξεργασίας είναι απαραίτητο και προηγείται πάντα της ανάλυσης των δεδομένων αφού σε πολύ μεγάλα σύνολα δεδομένων, ο λεγόμενος θόρυβος υπάρχει σε πολλές μορφές. Θόρυβος δεδομένων θεωρείται οτιδήποτε περιττό ή αποτελεί λανθασμένη πληροφορία για το συγκεκριμένο σύστημα. Και στις δύο περιπτώσεις η μη αφαίρεση του θορύβου μπορεί να οδηγήσει σε λανθασμένα αποτελέσματα και σε μεγάλες καθυστερήσεις το σύστημα, με συνέπεια την τελική αστοχία.

Ένας σημαντικός παράγοντας είναι η διαμόρφωση των δεδομένων στην μορφή που επιβάλλει η είσοδος του συστήματος. Σε διαφορετική περίπτωση το σύστημα ενδεχομένως να μην δεχτεί τα δεδομένα και να μη προχωρήσει ποτέ στην ανάλυσή τους ή θα έχει την ψευδαίσθηση ότι η είσοδος ήταν σωστή, χωρίς ποτέ να ειδοποιήσει για αυτή την αναστάτωση.

Στόχος μας είναι στο τέλος της προ-επεξεργασίας των δεδομένων να καταλήξουμε στην δημιουργία της εισόδου του αλγορίθμου του συστήματος σύστασης. Τα δεδομένα της εισόδου θα συγκεντρωθούν από διαφορετικές πηγές, όπως το EPG για πληροφορίες σχετικά με τα ζωντανά προγράμματα, το ZPG για πληροφορίες σχετικά με της μεταβολές του χρήστη στα ζωντανά προγράμματα όπου θα προκύψουν και τα σχετικά ratings των χρηστών, το VoD για πληροφορίες σχετικά με της επιλογές του χρήστη στο «on Demand» περιεχόμενο, το Catchup για πληροφορίες σχετικά με της επιλογές του χρήστη μέσα στο επόμενο 48ωρο από την ζωντανή προβολή του προγράμματος.

4.1. Βάση Δεδομένων

Το σύνολο των δεδομένων που χρησιμοποιήθηκαν στη παρούσα διατριβή προέρχονται από έναν από τους κορυφαίους τηλεπικοινωνιακούς ομίλους στον Κόσμο. Τα συγκεκριμένα δεδομένα δεν στηρίζονται σε προσωπικές πληροφορίες από τους χρήστες π.χ. ηλικία, φύλο, επάγγελμα. Αντίθετα περιέχονται πληροφορίες για το linear πρόγραμμα με τις μεταβολές που έκανε ο κάθε χρήστης σε αυτές, τις επιλογές αυτών πάνω στο on demand περιεχόμενο καθώς και τις επιλογές τους στην υβριδική υπηρεσία catchup που έχει περιγραφεί στο 3^{ο} κεφάλαιο.

Οι χρήστες αλληλεπιδρούν με το σύστημα IPTV μέσω του STB [Set Top Box], συνεπώς δεν μπορεί να προσδιοριστεί ποιος είναι πραγματικά μπροστά από την τηλεόραση. Κατά συνέπεια, το STB συλλέγει τη συμπεριφορά και τις προτιμήσεις ενός συνόλου των χρηστών (π.χ., η συνιστώσα της οικογένειας). Συνεπώς, όπου θα αναφερθούμε στα δεδομένα μας σε χρήστη είναι σας να αναφερόμαστε στο STB καθώς και το αντίστροφο, αφού έχουμε αντιστοιχία ένα προς ένα.

Το RDBMS (Relational Database Management System) που επιλέχτηκε να χρησιμοποιηθεί για τις ανάγκες της παραπάνω βάσης είναι η MySQL, που χρησιμοποιεί την Structured Query Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων. Ο κύριος λόγος που επιλέχθηκε είναι επειδή είναι ένα ελεύθερο σύστημα αλλά και επειδή είναι ευρύτατα διαδεδομένη τόσο στα UNIX συστήματα όσο και στον κόσμο των Windows. Φυσικά ήταν απαραίτητη η εγκατάσταση του πακέτου WAMP (Windows, Apache, MySQL Server, PHP) σε έναν υπολογιστή, ώστε να παίξει το ρόλο του SQL Server. Επίσης, για την εύκολη χρήση του MSQL Server

χρησιμοποιήθηκε η γραφική διεπαφή "MySQL Workbench", επιτρέποντας την διαχείριση του MySQL Server (πχ δημιουργία χρηστών, απονομή δικαιωμάτων σε χρήστες, κτλ.), την σχεδίαση σχεσιακών βάσεων δεδομένων (δημιουργία πινάκων, συσχετίσεων μεταξύ τους, κτλ.) και την δημιουργία και εκτέλεση SQL ερωτημάτων, όψεων, διαδικασιών, κτλ στην βάση δεδομένων.

Η συγκεκριμένη βάση δεδομένων αποτελείται από τέσσερις (4) βασικούς πίνακες ως εξής:

- **EPG:** Ο συγκεκριμένος πίνακας αποτελείται από στοιχεία που συγκροτούν τον ηλεκτρονικό οδηγό προγράμματος [Electronic Program Guide] για διάρκεια ενός έτους.
- **ZPG:** Ο συγκεκριμένος πίνακας εμπεριέχει όλες τις μεταβολές (Zappings), του ζωντανού προγράμματος των καναλιών, που πραγματοποιήθηκαν από κάθε χρήστη κατά την διάρκεια ενός έτους σε ανώνυμη μορφή.
- **VoD:** Ο συγκεκριμένος πίνακας αποτελείται από τα στοιχεία όλων των αντικειμένων VoD [Video on Demand] που επιλέχτηκαν από τους χρήστες κατά την διάρκεια ενός έτους σε ανώνυμη μορφή.
- **Catchup:** Ο συγκεκριμένος πίνακας αποτελείται από τα στοιχεία όλων των αντικειμένων Catchup που επιλέχτηκαν από τους χρήστες κατά την διάρκεια ενός έτους σε ανώνυμη μορφή.

4.1.1. Πίνακας EPG

Ο πίνακας EPG περιέχει τις εγγραφές του ηλεκτρονικού οδηγού προγράμματος για την διάρκεια ενός έτους. Το συνολικό πλήθος των εγγραφών είναι 797.427.
Η αρχική δομή του πίνακα EPG είναι η εξής:

EPG (programID, channelName, dateStart, timeStart, dateStop, timeStop, title, description, synopsis, rating, duration, catName, copyright)

EPG	
programID	Περιέχει έναν μοναδικό αριθμό που προσδιορίζει κάθε πρόγραμμα. Αποτελεί πρωτεύων κλειδί του πίνακα.
channelName	Αποτελεί το όνομα του εκάστοτε καναλιού.
dateStart	Ημερομηνία εκκίνησης του εκάστοτε προγράμματος.
timeStart	Προγραμματισμένη ώρα εκκίνησης του εκάστοτε προγράμματος.
dateStop	Ημερομηνία τερματισμού του εκάστοτε προγράμματος.
timeStop	Προγραμματισμένη ώρα τερματισμού του εκάστοτε προγράμματος.
title	Αποτελεί τον τίτλο του προγράμματος και κατά προέκταση το βασικό αντικείμενο του συστήματός μας.
description	Γενική περιγραφή του εκάστοτε προγράμματος.

synopsis	Συνοπτική περιγραφή του εκάστοτε προγράμματος.
rating	Περιέχει την αξιολόγηση με βάση το σύστημα διαβάθμισης περιεχομένου. π.χ. το κατάλληλο για όλες τις ηλικίες.
duration	Αποτελεί την διάρκεια του εκάστοτε προγράμματος και είναι καταχωρημένη σε λεπτά.
catName	Το όνομα της κατηγορίας που ανήκει το εκάστοτε πρόγραμμα π.χ. Reality.
copyright	Η χρονιά παραγωγής του εκάστοτε προγράμματος.

Τα σημαντικότερα στοιχεία του πίνακα, όπου και χρησιμοποιήθηκαν είναι programID που αποτελεί και το κλειδί του, το ChannelName, dateStart, dateStop, timeStart, timeStop, title, duration και το catName.

Οι στήλες dateStart & timeStart ενοποιήθηκαν ώστε να προκύψουν τιμές τύπου datetime ώστε μαζί με το ζεύγος πεδίων dateStop & timeStop να έχουμε 2 πεδία που να δηλώνουν την έναρξη και το τέλος του κάθε προγράμματος. Επιπλέον επιλέχτηκε η μορφή αναπαράστασης Unix Timestamp, ένα σύστημα που περιγράφει τον χρόνο, ως ο αριθμό των δευτερολέπτων που έχουν περάσει από 00:00:00 Πέμπτη, 1 Ιανουαρίου 1970.

Τέλος, το πεδίο title πέρασε αρκετές φάσεις επεξεργασίας ώστε να αποτελέσει και το τελικό αντικείμενο του συστήματος.

Πιο συγκεκριμένα:

- Αφαιρέθηκαν παρενθέσεις, διπλά πολλαπλά κενά, κόμματα ή άχρηστοι ειδικοί χαρακτήρες ώστε να μην εμποδίζουν στην σωστή αναγνώριση του αντικειμένου. Παράδειγμα το Play It Again, Sam υπήρχε με κόμμα(,) και χωρίς κόμμα με αποτέλεσμα η μη διόρθωσή του να οδηγούσε σε 2 ίδια αντικείμενα που το σύστημα θα τα αναγνώριζε για διαφορετικά.
- Αφαιρέθηκαν γράμματα και αριθμοί που δήλωναν αριθμό σεζόν ή αριθμό επεισοδίου ώστε να ομαδοποιηθούν σωστά τα αντικείμενα. Παράδειγμα το CSI: Miami (S1E05) μετατράπηκε σε CSI: Miami.
- Διορθώθηκαν τίτλοι που είχαν πρώτα το ουσιαστικό και μετά το άρθρο σε ορισμένες περιπτώσεις. Παράδειγμα το Insider, The μετατράπηκε σε The Insider.

4.1.2. Πίνακας ZPG

Ο πίνακας ZPG περιέχει τις εγγραφές κάθε εναλλαγής καναλιού (zapping) του εκάστοτε χρήστη για την διάρκεια ενός έτους. Το συνολικό πλήθος των εγγραφών είναι 294.575.427.

Η αρχική δομή του πίνακα ZPG είναι η εξής:

ZPG (zpgID, subID, timeStart, duration, channelName, year, month, week, day)

ZPG	
zpgID	Περιέχει έναν μοναδικό αριθμό που προσδιορίζει κάθε εναλλαγή του καναλιού. Αποτελεί πρωτεύων κλειδί του πίνακα.
subID	Αποτελεί τον μοναδικό αριθμός χρήστη.
timeStart	Ώρα εκκίνησης της εναλλαγής της εκάστοτε αλλαγής.
duration	Διάρκεια παραμονής στο ίδιο κανάλι χωρίς να υπάρξει εναλλαγή σε δευτερόλεπτα.
channelName	Αποτελεί τον τίτλο του καναλιού.
year	Το έτος που πραγματοποιήθηκε η εναλλαγή.
month	Ο μήνας που πραγματοποιήθηκε η εναλλαγή.
week	Ο αύξων αριθμός της εβδομάδας που πραγματοποιήθηκε η εναλλαγή.
day	Η ημέρα που πραγματοποιήθηκε η εναλλαγή.

Χρειάστηκαν να γίνουν αρκετές αλλαγές ώστε να προκύψει ο πίνακας στην επιθυμητή κατάσταση.

Πρώτα από όλα υπολογίστηκε ο χρόνος τερματισμού κάθε θέασης του εκάστοτε καναλιού με την βοήθεια της διάρκειας (πεδίο duration). Ο υπολογισμός αυτός έδωσε την δυνατότητα να υπολογιστούν τα datetime λήξης και έναρξης της εναλλαγής και να προκύψουν τα unix timestamp, για την πληρότητα και την ομοιομορφία με το EPG πίνακα.

```
ALTER TABLE ZPG ADD timeStop;
UPDATE ZPG
SET timeStop = addtime(timeStart,
SEC_TO_TIME(duration));
```

Η κάθε εγγραφή του πίνακα περιέχει το κανάλι, την διάρκεια που παρέμεινε ο χρήστης σε αυτό καθώς και την χρονική στιγμή εκκίνησης και τον τερματισμού της εκάστοτε θέασης. Συνεπώς, ήταν απαραίτητο να διαπιστωθεί το συγκεκριμένο πρόγραμμα που έπαιζε το κανάλι με βάση τον πίνακα EPG.

Στην συνέχεια ήταν απαραίτητο το group των επαναληπτικών εναλλαγών και του υπολογισμού του νέου duration που προκύπτει. Δηλαδή όταν ένας χρήστης παρακολουθούσε ένα συγκεκριμένο πρόγραμμα, μπορεί αυτό το πρόγραμμα να το αλλάξει για λίγο και να επανέλθει. Το αποτέλεσμα είναι να δημιουργηθεί μία αρχική εγγραφή στο 1ο κανάλι για το Α πρόγραμμα, μια δεύτερη εγγραφή για το 2ο κανάλι

και ένα Β πρόγραμμα και μια 3η εγγραφή πάλι για το ίδιο κανάλι και πρόγραμμα (δηλαδή 1ο κανάλι Α πρόγραμμα). Αυτό έχει ως συνέπεια την δημιουργία πολλών ίδιων εγγραφών ως προς το πρόγραμμα το κανάλι και τον χρήστη την ίδια ημέρα. Μετά την απαραίτητη ομαδοποίηση των παραπάνω εγγραφών υπολογίστηκε εκ νέου το συνολικό πλέον duration της κάθε θέασης αθροιστικά. Με αυτόν τον τρόπο γνωρίζουμε τον συνολικό χρόνο που κατανάλωσε ο εκάστοτε χρήστης για το κάθε πρόγραμμα συνολικά, ανεξάρτητα από τις εναλλαγές που έκανε.

Η συγκεκριμένη βάση δεδομένων δεν περιείχε κάποιον άμεσο τρόπο βαθμολόγησης του κάθε προγράμματος. Συνεπώς στα πλαίσια του συστήματος σύστασης που κατασκευάστηκε έγινε η παραδοχή ότι ο χρήστης βαθμολογεί / αξιολογεί το κάθε αντικείμενο βάσει του λόγου της διάρκειας που παρακολουθεί ένα πρόγραμμα προς την συνολική διάρκεια του προγράμματος. Με αυτόν τον τρόπο έμμεσα εκμαιεύτηκε η βαθμολογία (rating) του χρήστη στο εκάστοτε πρόγραμμα. Για τους λόγους της παραπάνω παραδοχής, υπολογίστηκε τόσο ο συνολικός χρόνος διάρκειας του κάθε προγράμματος όσο και ο λόγος της διάρκειας θέασης προς αυτόν τον συνολικό χρόνο.

Επιπρόσθετα δημιουργήθηκε ένα πεδίο με την ονομασία status όπου είχε δύο καταστάσεις, idle και ok. Στην περίπτωση που ο χρήστης είχε πάνω από 4,5 ώρες να πραγματοποιήσει μια εναλλαγή (zapping) το status ήταν idle και σε διαφορετική περίπτωση ok. Με αυτό τον τρόπο μπορούσε να γίνει αντιληπτό αν κάποιος χρήστης ήταν ενεργός (ok) και έβλεπε τηλεόραση ή ανενεργός (idle) και ενδεχομένως με ξεχασμένο τον STB για πολλές ώρες ανοιχτό χωρίς να βλέπει.

Τέλος, υπολογίστηκε μοναδικός αριθμός για κάθε αντικείμενο του συστήματος για την ευκολία της αναφοράς σε αυτό.

4.1.3. Πίνακας VoD

Ο πίνακας VoD περιέχει τις εγγραφές όλων των αντικειμένων VoD [Video on Demand] που επιλέχτηκαν από τους χρήστες κατά την διάρκεια ενός έτους. Το συνολικό πλήθος των εγγραφών είναι 105.166.

Ο δομή του πίνακας VoD είναι η εξής:

VoD (vodID, subID, dateStart, timeStart, title, price, duration, year, month, week, day, hour, catName)

VoD	
vodID	Περιέχει έναν μοναδικό αριθμό που προσδιορίζει κάθε επιλογή on demand περιεχομένου. Αποτελεί πρωτεύων κλειδί του πίνακα.
subID	Αποτελεί τον μοναδικό αριθμός χρήστη.
dateStart	Ημερομηνία εκκίνησης on Demand περιεχομένου από τον εκάστοτε χρήστη.
timeStart	Ώρα εκκίνησης on Demand περιεχομένου από τον εκάστοτε χρήστη.
title	Αποτελεί τον τίτλο του on Demand περιεχομένου και παράλληλα το αντικείμενο του συστήματος.

price	Αναφέρεται η τιμή κόστους θέασης.
duration	Διάρκεια on Demand περιεχομένου.
year	Το έτος που πραγματοποιήθηκε η θέαση του περιεχομένου.
month	Ο μήνας που πραγματοποιήθηκε η θέαση του περιεχομένου.
week	Η εβδομάδα που πραγματοποιήθηκε η θέαση του περιεχομένου.
day	Η ημέρα που πραγματοποιήθηκε η θέαση του περιεχομένου.
hour	Η ώρα που πραγματοποιήθηκε η θέαση του περιεχομένου.
catName	Η κατηγορία του on Demand περιεχομένου.

Έγινε η μετατροπή του dateStart & timeStart σε Unix timestamp ώστε να υπάρχει ομοιομορφία με τους υπόλοιπους πίνακες.

Το πεδίο title επεξεργάστηκε κατάλληλα όπως το αντίστοιχο του πίνακα EPG. Δηλαδή αφαιρέθηκαν παρενθέσεις, διπλά πολλαπλά κενά, κόμματα, γράμματα και αριθμοί που δήλωναν αριθμό σεζόν ή αριθμό επεισοδίου, τίτλοι που είχαν πρώτα το ουσιαστικό και μετά το άρθρο κλπ.

Αφαιρέθηκε κάθε εγγραφή με αντικείμενα που δεν υπήρχαν στο linear πρόγραμμα για λόγους πληρότητας.

Τέλος αφαιρέθηκαν όλες οι εγγραφές που είχαν duration θέασης του on Demand περιεχομένου πολύ μικρό, της τάξεως κάτω του 10% του συνολικού χρόνου, διότι θεωρήθηκε ότι δεν επιλέχθηκε το συγκεκριμένο πρόγραμμα θελημένα από τον χρήστη, γεγονός που αποτελεί την επιλογή μη έγκυρη.

4.1.4. Πίνακας CTP

Ο πίνακας CTP περιέχει τις εγγραφές όλων των αντικειμένων CTP [Catchup] που επιλέχτηκαν από τους χρήστες κατά την διάρκεια ενός έτους. Ο χρήστης έχει την δυνατότητα να δει ένα πρόγραμμα ακόμη και 48 ώρες μετά την ζωντανή μετάδοσή του. Το συνολικό πλήθος των εγγραφών είναι 4.459.049

Ο δομή του πίνακας CTP είναι η εξής:

CTP (ctpID, subID, dateStart, timeStart, title, price, duration, year, month, week, day, hour, catName)

CTP	
ctpID	Περιέχει έναν μοναδικό αριθμό που προσδιορίζει κάθε επιλογή on demand περιεχομένου. Αποτελεί πρωτεύων κλειδί του πίνακα.
subID	Αποτελεί τον μοναδικό αριθμός χρήστη.
dateStart	Ημερομηνία εκκίνησης on Demand περιεχομένου από τον εκάστοτε χρήστη.

timeStart	Ώρα εκκίνησης on Demand περιεχομένου από τον εκάστοτε χρήστη.
title	Αποτελεί τον τίτλο του on Demand περιεχομένου και παράλληλα το αντικείμενο του συστήματος.
price	Αναφέρεται η τιμή κόστους θέασης.
duration	Διάρκεια on Demand περιεχομένου.
year	Το έτος που πραγματοποιήθηκε η θέαση του περιεχομένου.
month	Ο μήνας που πραγματοποιήθηκε η θέαση του περιεχομένου.
week	Η εβδομάδα που πραγματοποιήθηκε η θέαση του περιεχομένου.
day	Η ημέρα που πραγματοποιήθηκε η θέαση του περιεχομένου.
hour	Η ώρα που πραγματοποιήθηκε η θέαση του περιεχομένου.
catName	Η κατηγορία του on Demand περιεχομένου.

4.2. Μορφή Εισόδου Δεδομένων

Για την τελική μορφή των δεδομένων αντιστοιχήθηκαν τα IDs των χρηστών, των αντικειμένων και της κατηγορίας σε μοναδικό αύξοντα αριθμό, ξεκινώντας πάντα από το 1. Μετά την διαμόρφωση των δεδομένων προέκυψαν 3 πίνακες που αποτελούσαν και τα δεδομένα εισόδου του συστήματος σύστασης.

Οι πίνακες είναι οι εξής:

- ZPG_INPUT [Linear]
- CTP_INPUT [Catchup]
- VOD_INPUT [Video on Demand]

4.2.1. Πίνακας ZPG_INPUT [Linear]

Η δομή του πίνακα είναι : ZPG_INPUT (subID_int, itemID_int, catID_int, rating, UnixTimestamp)

Ο πίνακας συνολικά αποτελείται από 219.767.776 εγγραφές.

ZPG_INPUT			
Όνομα πεδίου	min	MAX	Περιγραφή
subID_int	1	57826	Ο μοναδικός αριθμός χρήστη.

itemID_int	1	15802	Ο μοναδικός αριθμός του αντικειμένου.
catID_int	1	15	Ο μοναδικός αριθμός κατηγορίας του αντικειμένου.
rating	0	1	Αποτελεί την βαθμολογία του χρήστη προς το εκαστοτε αντικείμενο.
UnixTimestamp	1056991201	1088526758	Αποτελεί το Unix timestamp της χρονικής στιγμής που αποδόθηκε η βαθμολογία.

4.2.2. Πίνακας CTP_INPUT [Catchup]

Η δομή του πίνακα είναι : CTP_INPUT (subID_int, itemID_int, catID_int, UnixTimestamp)

Ο πίνακας συνολικά αποτελείται από 4.459.049 εγγραφές.

CTP_INPUT			
Όνομα πεδίου	min	MAX	Περιγραφή
subID_int	1	43343	Ο μοναδικός αριθμός χρήστη.
itemID_int	1	7845	Ο μοναδικός αριθμός του αντικειμένου.
catID_int	1	15	Ο μοναδικός αριθμός κατηγορίας του αντικειμένου.
UnixTimestamp	1056991201	1088526758	Αποτελεί το Unix timestamp της χρονικής στιγμής της προτίμησης.

4.2.3. Πίνακας VoD_INPUT [Video on Demand]

Η δομή του πίνακα είναι : VoD_INPUT (subID_int, itemID_int, catID_int, UnixTimestamp)

Ο πίνακας συνολικά αποτελείται από 4.459.049 εγγραφές.

CTP_INPUT			
Όνομα πεδίου	min	MAX	Περιγραφή
subID_int	1	13638	Ο μοναδικός αριθμός χρήστη.
itemID_int	1	401	Ο μοναδικός αριθμός του αντικειμένου.
catID_int	1	15	Ο μοναδικός αριθμός κατηγορίας του αντικειμένου.
UnixTimestamp	1056991201	1088526758	Αποτελεί το Unix timestamp της χρονικής στιγμής της προτίμησης.

5. Περιγραφή Υλοποίησης

Στο μοντέλο που έχει υλοποιηθεί και παρουσιάζεται παρακάτω βασίζεται στον αλγόριθμο SVD++ [12] με ενσωμάτωση της παραμέτρου του χρόνου. Ετήσιά, στον βασικό εκτιμητή των βαθμολογιών του χρήστη ενσωματώνεται η παράμετρος του χρόνου με τη χρήση ενός, κατά τμήματα, συνεχούς μοντέλου κατακερματίζοντας για κάθε χρήστη τον άξονα του χρόνου σε ένα προκαθορισμένο αριθμό σημείων.

Για να επιτευχθεί αυτό η γραμμή του χρόνου των αλληλεπιδράσεων κάθε χρήστη που καλύπτει το συνολικό χρονικό διάστημα $[t^u_{min}, t^u_{max}]$ κατανέμεται σε έναν αριθμό από K_u ίσα διαστήματα έτσι ώστε:

Όπου $|\Delta t_1|=|\Delta t_2|=\ldots=|\Delta t_{Ku+1}|=\frac{t^u_{max}-t^u_{min}}{n_u}$ και n_u είναι μια παράμετρος διαστήματος η οποία θα καθοριστεί πειραματικά για κάθε χρήστη.

Το βασικό μοντέλο χρήστη θεωρείτε ότι είναι γραμμικό με κάθε χρονικό διάστημα να εντάσσεται σε μια φόρμα λειτουργίας που μπορεί να μορφοποιηθεί ως εξής:

$$b_u(t) = b_u^0 + \Delta t * \sum_{k=1}^{k-1} (b_u^k + b_u^k * (t - t_k)) \qquad (1)$$

$(b_u^0, b_u^k : k \in [k_{u+1}] \in \mathbb{R})$

για $t \in [t_k, t_{k+1}]$

και $k \in [1 \ldots k_u + 1]$

Για να εκτελεστούν καλλίτερα και ευκολότερα οι υπολογισμοί έγινε εισαγωγή των παρακάτω ποσοτήτων:

$d_{evu}(t) = t - t_k$, for $t \in [t_k, t_{k+1}]$ for $k \in [1 \ldots k_u + 1]$ (2)

$\underline{b^u} = [\, b_u^1, b_u^2 \ldots b_u^{k_u+1} \,]$ τέτοιο ώστε $\underline{b^u} \in \mathbb{R}^{k_u+1}$ (3)

$\underline{prev_u(t)} \in \mathbb{R}^{k_u+1}$ τέτοιο ώστε:

$\underline{prev_u(t)} = [prev_u^k(t)]$ όπου $prev_u^k(t) = \begin{cases} 1, & t > t_{k+1} \\ 0, & \text{αλλιώς} \end{cases}, \forall\, k \in [k_u + 1]$ (4)

Το $prev_u(t)$ είναι ένα δυαδικό διάνυσμα που αναφέρεται στα χρονικά διάστημα που προηγούνται της τρέχουσας χρονικής στιγμή (t).

$\underline{curr_u(t)} \in R^{k_u+1}$ τέτοιο ώστε :

$\underline{curr_u(t)} = [curr_u^k(t)]$ όπου $curr_u^k(t) = \begin{cases} 1, & \text{αν } t \in [t_k, t_{k+1}] \\ 0, & \text{αλλιώς} \end{cases}, \forall\, k \in [k_u+1]$ (5)

Το $curr_u(t)$ είναι ένα δυαδικό διάνυσμα που αναφέρεται στο χρονικό διάστημα που εμπεριέχει την τρέχουσα χρονική στιγμή (t).

Σε αυτό το πλαίσιο ο βασικός εκτιμητής (baseline predictor) για την προτίμηση του χρήστη μπορεί να διαμορφωθεί σε ένα διάνυσμα ως [13,14,15]:

$$b_u(t) = b_u^o + \Delta t * \left(\underline{prev_u(t)} * \underline{b^r}^T\right) + dev_u(t) * \left(\underline{curr_u(t)} * \underline{b^r}^T\right) \quad (6)$$

Το μοντέλο που περιγράφεται από την εξίσωση (6) μπορεί να αυξηθεί λαμβάνοντας υπόψη συγκεκριμένες χρονικές στιγμές που σχετίζεται με την προτίμηση έτσι ώστε να μπορούμε να γράψουμε ότι:

$$b_u(t) = b_u^o + \Delta t * \left(\underline{prev_u}(t) * \underline{b^r}^T\right) + dev_u(t) * \left(\underline{curr_u(t)} * \underline{b^r}^T\right) + b_{u,t} \quad (7)$$

Για την ώρα ο βασικός εκτιμητής πρόβλεψης θα πρέπει να χρησιμοποιεί το ακόλουθο μοντέλο

$$b_i(t) = b_i + b_{i,BIN(t)} + b_{i,\,DAY(t)} \quad (8)$$

όπου $b_i, b_{i,BIN(t)}, b_{i,\,DAY(t)} \in \mathbb{R}$

Έχοντας κατά νου ότι το χρονοδιάγραμμα για κάθε αντικείμενο είναι το ίδιο που καλύπτει το πλήρες φάσμα του χρόνου όλων των αλληλεπιδράσεων των χρηστών $t \in [t_{min}, t_{max}]$

Η εξίσωση (8) μπορεί να αυξηθεί κατά τέτοιο τρόπο, ώστε να συμπεριληφθεί η πρόσθετη χρονική περίοδο για συγκεκριμένα γεγονότα, καθώς επεξεργαζόμαστε ένα σύστημα συστάσεων για τηλεόραση έτσι ώστε

$$b_i(t) = b_i + b_{i,BIN(t)} + b_{i,\,DAY(t)} + b_{i,SEASON(t)} + b_{i,TIMEZONE(t)}$$

Η γνώση του χρόνου μπορεί επίσης να εισαχθεί εντός των διανυσματικών συντελεστών αξιολόγησης των χρηστών με τον ίδιο τρόπο που έγινε για τον βασικό εκτιμητή προβλέψεων χρήστη. Προκειμένου να προκύψει ο συντελεστής αντιστοίχισης και υποθέτοντας την ίδια χρονική επέκταση για κάθε χρήστη μπορούμε να γράψουμε την παρακάτω εξίσωση:

$$\underline{P_u}(t) = \underline{P_u^o} + \Delta t * \sum_{n=0}^{k-1} \left(\underline{P_u^n} * \Delta t + \underline{P_u^k} * (t - t_k) \right) \qquad (10)$$

$(\underline{P_u^o}, \underline{P_u^k} \in \mathbb{R}^f)$

για $k \in [t_k, t_{k+1}]$ και $k \in [k_u+1]$

Για να εκτελεστούν οι βασικοί υπολογισμοί και να χρησιμοποιηθεί ένας πιο ολοκληρωμένος συμβολισμός εισάγουμε μια σειρά δεικτών έτσι ώστε:

$\underline{P}(u,t)$: είναι μια σειρά από δείκτες χρονικού διαστήματος που προηγούνται των χρονικών διαστημάτων που περιέχουν την συγκεκριμένη χρονική στιγμή (t) για τον χρήστη (u). (11)

$\underline{C}(u,t)$: είναι ο δείκτης του χρονικού διαστήματος που περιέχει την τρέχουσα χρονική στιγμή (t) για τον χρήστη (u). (12)

Σύμφωνα με τους ορισμούς (11), (12), η εξίσωση (10) μπορεί να ξαναγραφεί με τον ακόλουθο τρόπο:

$$\underline{P_u}(t) = \underline{P_u^o} + \sum_{j \in P_{u,t}} \underline{P_u^j} * \Delta t + \sum_{j \in C_{u,t}} \underline{P_u^j} * dev_u(t) \qquad (13)$$

Ενσωματώνοντας τον στιγμιαίο χρόνο των ειδικών προτιμήσεων η παραπάνω εξίσωση μπορεί να γραφεί:

$$\underline{P_u}(t) = \underline{P_u^o} + \sum_{j \in P_{u,t}} \underline{P_u^j} * \Delta t + \sum_{j \in C_{u,t}} \underline{P_u^j} * dev_u(t) + \underline{P_{u,t}} \qquad (14)$$

Το μοντέλο, εκτός από την ενσωμάτωση της χρονικής μεταβολής των προτιμήσεων του χρήστη ενσωματώνει και δύο τύπους έμμεσης ανατροφοδότησης. Τα ανατροφοδοτούμενα αυτά στοιχεία είναι τα γεγονότα catchup και τα συμβάντα VOD που σχετίζονται με την αλληλεπίδραση του χρήστη με την τηλεόραση.

Έστω $N_{(u)}^{catch}$ το σύνολο των αντικειμένων που έγιναν catched up από έναν συγκεκριμένο χρήστη και έστω $N_{(u)}^{vod}$ το σύνολο των αντικειμένων που έγιναν VOD από έναν συγκεκριμένο χρήστη.

Τα αντίστοιχα διανύσματα των σχετικών στοιχείων θα είναι:

$$Y_j^{catch}, Y_j^{vod} \in \mathbb{R}^f \qquad (15)$$

Έχοντας κατά νου όλους τους προηγούμενους ορισμούς, ο συντελεστής του συνολικού γνωστού χρόνου του μοντέλου παράγει μια εκτίμηση $\hat{r}_{ui}(t)$ σύμφωνα με την ακόλουθη εξίσωση:

$$\hat{r}_{ui}(t) = \mu + b_u(t) + b_i(t) + \underline{q_i}^T \left\{ \underline{P}_{u,t} + \left|N_{(u)}^{catch}\right|^{-\frac{1}{2}} \sum_{j \epsilon N_{(u)}^{catch}} Y_j^{catch} + \left|N_{(u)}^{VOD}\right|^{-\frac{1}{2}} \sum_{j \epsilon N_{(u)}^{catch}} Y_j^{VOD} \right\} \quad (16)$$

Διατυπώνοντας τη συνάρτηση κόστους ελαχιστοποιώντας τις απαιτήσεις πρέπει να ελαχιστοποιηθεί απαιτεί τον καθορισμό των κατάλληλων όρων κανονικοποιησης για κάθε παράμετρο, όπως

$(b_u^0)^2$: ο όρος που σχετίζεται με την προτίμηση του χρήστη

$\|\underline{b}^r\|^2$: για το διάνυσμα προτίμησης του χρήστη

$b_{u,t}^2$: ο όρος που είναι συσχετισμένος με την στιγμιαία προτίμηση ενός συγκεκριμένου χρήστη.

b_i^2 : για τον όρο προτίμησης στοιχείων

$b_{i,\,BIN(t)}^2$: ο όρος που είναι συσχετισμένος με ένα συγκεκριμένο χρονικό διάστημα (στον άξονα χρόνου όλων των αντικειμένων)

$b_{i,\,DAY(t)}^2$: ο όρος που είναι συσχετισμένος με μια συγκεκριμένη μέρα της εβδομάδας

$\|P_u^o\|^2$: ο όρος που σχετίζεται με το σταθερό διάνυσμα του χρήστη.

$\sum_{j \in P(u,t)} \|P_u^j\|^2$ $και$ $\sum_{j \in C(u,t)} \|P_u^j\|^2$: Για το διάνυσμα χρήστη, όρος που σχετίζεται με τον εσωτερικό χρόνο.

$\|P_{u,t}\|^2$: όρος για το διάνυσμα χρήστη που σχετίζεται με τον όρο του στιγμιαίου χρόνου.

Ελαχιστοποιώντας την συνάρτηση σφάλματος έχουμε:

$$E = \sum_{(u,i,t) \epsilon k} \frac{1}{2} \left(r_{ui}(t) - \hat{r}_{ui}(t)\right)^2 + \frac{\lambda}{2} \left(b_u^{0\,2} + \left\|\underline{\hat{b}}^r\right\|^2 + b_{u,t}^2 + b_i^2 + b_{i,BIN\,(t)}^2 + b_{i,DAY(t)}^2 + \cdots + \left\|P_u^o\right\|^2 + \sum_{j \epsilon N_{(u)}^{catch}} \left\|\underline{Y}_j^{catch}\right\|^2 + \sum_{j \epsilon N_{(u)}^{VOD}} \left\|\underline{Y}_j^{VOD}\right\|^2 + \cdots + \sum_{j \epsilon P(u,t)} \left\|\underline{P}_u^j\right\|^2 + \sum_{j \epsilon C(u,t)} \left\|\underline{P}_u^j\right\|^2 \right) \quad (17)$$

$\underline{\hat{b}}^{r} = \underline{prev}_{u}(t) + \underline{curr}_{u}(t) \odot \underline{b}^{r}$ $\mu\varepsilon$ $\odot$ element wise

Όπου το k αποθηκεύει όλες τις τριάδες για τις οποίες υπάρχουν γνωστές αξιολογήσεις.

Λαμβάνοντας υπόψη αυτό η εξίσωση (17) μπορεί να γραφτεί ξανά ως:

$$E = \sum_{(u,i,t)\in k} Euit \quad (18) \quad \text{και}$$

$$e_{uit} = r_{ui}(t) - \hat{r}_{ui}(t) \quad (19)$$

μ: είναι η συνολική μέση τιμή της βαθμολογίας [σταθερά] (20)

Η ελαχιστοποίηση του λειτουργικού κόστους στην εξίσωση (18) μπορεί να διεξαχθεί μέσω της αξιοποίησης του αλγορίθμου stochastic gradient descent. Για το λόγω αυτό χρειαζόμαστε να υπολογίσουμε τις παρακάτω παραγώγους.

$\odot\nabla\, b_u^0 Euit \quad \odot\nabla\, \underline{b}^{r} Euit \quad \odot\nabla\, b_{u,t} Euit \quad \forall\, (u,i,t)\in k$

$\odot\nabla\, b_i Euit \quad \odot\nabla\, b_{i,BIN(t)} \quad \odot\nabla\, b_{i,DAY(t)} \quad \odot\nabla\, \underline{P}_u^{o} Euit \quad \odot\nabla\, \underline{P}_{u,t} Euit$

$\forall\, j\in P_{(u,t)} : \nabla\, \underline{P}_u^{j} Euit$ (με, $j\neq 0$)

$\forall\, j\in C_{(u,t)} : \nabla\, \underline{P}_u^{j} Euit$ (με, $j\neq 0$ και $|(u,t)|=1$)

$\forall\, j\in N_{(u)}^{catch} : \nabla\, \underline{Y}_j^{catch} Euit$

$\forall\, j\in N_{(u)}^{VOD} : \nabla\, \underline{Y}_j^{VOD} Euit$

5.1. Υπολογισμοί παραγώγων

Λαμβάνοντας υπόψη ότι η συνάρτηση κανονικοποίησης που αναφέρεται στη παράμετρο σε σχέση με την παράμετρο

b_u^0 είναι $R_{b_u^0}(b_u^0) = {b_u^0}^2 * \frac{\lambda}{2} + C_{b_u^0}$ (21)

Μπορούμε να γράψουμε ότι:

$$E = \frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + {b_u^0}^2 * \frac{\lambda}{2} + C_{b_u^0} \quad (22)$$

Όπου μας δίνει ότι:

$$\nabla b_u^0 Euit = \frac{\partial}{\partial b_u^0}[\frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + b_u^{0\,2} * \frac{\lambda}{2} + C_{b_u^0}] \Rightarrow$$

$$\nabla b_u^0 Euit = (r_{ui}(t) - \hat{r}_{ui}(t)) \frac{\partial}{\partial b_u^0}(r_{ui}(t) - \hat{r}_{ui}(t)) + \lambda b_u^0 \Rightarrow$$

$$\nabla b_u^0 Euit = -e_{uit} + \lambda b_u^0 \quad (23)$$

Η συνάρτηση κανονικοποίησης σε σχέση με τη διανυσματική παράμετρο $\underline{b}^r$ είναι:

$$R_{\underline{b}^r}(\underline{b}^r) = \frac{\lambda}{2}\left\|\underline{\hat{b}}^r\right\|^2 + C_{\underline{b}^r} \Rightarrow R_{\underline{b}^r}(\underline{b}^r) = \frac{\lambda}{2}(\underline{\hat{b}}^r)^T \underline{\hat{b}}^r + C_{\underline{b}^r}$$

Η οποία δίνει:

$$\nabla \underline{b}^r Euit = \frac{\partial}{\partial \underline{b}^r}[\frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + \frac{\lambda}{2}(\underline{\hat{b}}^r)^T \hat{b}^r + C_{\underline{b}^r}] \Rightarrow$$

$$\nabla \underline{b}^r Euit = (r_{ui}(t) - \hat{r}_{ui}(t)) \frac{\partial}{\partial \underline{b}^r}(r_{ui}(t) - \hat{r}_{ui}(t)) + \lambda \underline{\hat{b}}^r \quad (24)$$

Αλλά έχουμε ότι:

$$\frac{\partial}{\partial \underline{b}^r}[r_{ui}(t) - \hat{r}_{ui}(t)] = -\frac{\partial}{\partial \underline{b}^r}[\Delta t * \underline{prev_u}(t) * \underline{b}^{r\,T} + \underline{dev_u}(t) * \underline{curr_u(t)} * \underline{b}^{r\,T}] =$$

$$-\Delta t * \underline{prev_u}(t) - dev_u(t) * curr_u(t) \quad (25)$$

Συνδυάζοντας τις εξισώσεις (24), (25) έχουμε :

$$\nabla \underline{b}^r Euit = -e_{uit} * [\Delta t * \underline{prev_u}(t) + \underline{dev_u}(t) * \underline{curr_u}(t)] + \lambda * (\underline{prev_u}(t) + \underline{curr_u}(t)) \odot \underline{b}^r \quad (26)$$

Η συνάρτηση κανονικοποίησης σε σχέση με την παράμετρο $b_{u,t}$ είναι

$$R_{b_{u,t}}(b_{u,t}) = b_{u,t}^2 * \frac{\lambda}{2} + C_{b_{u,t}} \quad (27)$$

Έτσι μπορούμε να γράψουμε ότι

$$E_{uit} = \frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + \frac{\lambda}{2} b_{u,t}^2 + C_{b_{u,t}} \quad (28)$$

Επομένως παίρνουμε ότι:

$$\nabla b_{u,t} Euit = \frac{\partial}{\partial b_{u,t}}\left[\frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + \frac{\lambda}{2} b_{u,t}^2 + C_{b_{u,t}}\right] \Rightarrow$$

$$\nabla b_{u,t} Euit = (r_{ui}(t) - \hat{r}_{ui}(t)) \frac{\partial}{\partial b_{u,t}} [(r_{ui}(t) - \hat{r}_{ui}(t)) + \lambda b_{u,t}] \Rightarrow$$

$$\nabla b_{u,t} Euit = -e_{uit} + \lambda b_{u,t} \quad (29)$$

Εφαρμόζοντας την ίδια λογική προκύπτουν οι παρακάτω εξισώσεις:

$$\nabla b_i Euit = -e_{uit} + \lambda b_i \quad (30)$$

$$\nabla b_{i,BIN(t)} = -e_{uit} + \lambda b_{i,BIN(t)} \quad (31)$$

$$\nabla b_{i,DAY(t)} = -e_{uit} + \lambda b_{i,DAY(t)} \quad (32)$$

Συνδέοντας την παράμετρο $\underline{P_u^0}$ μπορούμε να γράψουμε :

$$Euit\left(\underline{P_u^0}\right) = \frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + \frac{\lambda}{2}\left\|P_u^0\right\|^2 + C_{\underline{P_u^0}} \quad (33)$$

όπου:

$$\hat{r}_{ui}\left(\underline{P_u^0}, t\right) = \underline{q_i^T}\ \underline{P_u^0} + \hat{C}_{\underline{P_u^0}} \quad (34)$$

Το οποίο δίνει

$$\nabla_{\underline{P_u^0}} Euit = \frac{\partial}{\partial \underline{P_u^0}}\left[\frac{1}{2}(r_{ui}(t) - \hat{r}_{ui}(t))^2 + \frac{\lambda}{2}\left\|P_u^0\right\|^2 + C_{\underline{P_u^0}}\right] \Rightarrow$$

$$\nabla_{\underline{P_u^0}} Euit = (r_{ui}(t) - \hat{r}_{ui}(t)) \frac{\partial}{\partial \underline{P_u^0}}[(r_{ui}(t) - \hat{r}_{ui}(t))] + \lambda \underline{P_u^0} \Rightarrow$$

$$\nabla_{\underline{P_u^0}} Euit = -e_{uit} \frac{\partial}{\partial \underline{P_u^0}}[\underline{q_i^T}\ \underline{P_u^0} + \hat{C}_{\underline{P_u^0}}] + \lambda \underline{P_u^0} \Rightarrow$$

$$\nabla_{\underline{P_u^0}} Euit = -e_{uit} * \underline{q_i} + \lambda \underline{P_u^0} \quad (35)$$

Εφαρμόζοντας την ίδια λογική παράγονται οι παρακάτω εξισώσεις:

$$\nabla_{\underline{P_{u,t}}} Euit = -e_{uit} * \underline{q_i} + \lambda \underline{P_{u,t}} \quad (36)$$

Και

$\forall\ j \in P_{(u,t)}$: $\nabla\ \underline{P_u^j} Euit$ μπορούν να παραχθούν οι παρακάτω εξισώσεις:

$$\nabla_{\underline{P_u^j}} Euit = -e_{uit} * \Delta t * \underline{q_i} + \lambda \underline{P_u^j} \quad (37)$$

$$\nabla_{\underline{P_u^j}} Euit = -e_{uit} * dev_u(t) * \underline{q_i} + \lambda \underline{P_u^j} \quad (38)$$

Για τα σχετικά διανύσματα της έμμεσης ανάδρασης μπορούμε να γράψουμε ότι:

$$\forall\, j \in N_{(u)}^{catch} : \nabla \underline{Y}_j^{catch} Euit = -|N_{(u)}^{catch}|^{-\frac{1}{2}} * e_{uit} * \underline{q_i} + \lambda \underline{Y}_j^{catch} \quad (39)$$

$$\forall\, j \in N_{(u)}^{VOD} : \nabla \underline{Y}_j^{VOD} Euit = -|N_{(u)}^{VOD}|^{-\frac{1}{2}} * e_{uit} * \underline{q_i} + \lambda \underline{Y}_j^{VOD} \quad (40)$$

Το μοντέλο μας θα πρέπει να συγκριθεί με το αρχικό γραμμικό μοντέλο για τον βασικό εκτιμητή χρήστη, σύμφωνα με το οποίο έχουμε ότι

$$b_u(t) = b_u + a_u * dev_u(t) + b_{u,t} \quad (41)$$

Όπου: b_u, a_u and $b_{u,t} \in \mathbb{R}$ και $dev_u(t) = sign(t - t_u) * |t - t_u|^B$ (42)

Με $t_u = \frac{t_u^{min} - t_u^{max}}{2}$ (43)

Σε αυτό το πλαίσιο οι εξισώσεις (22), (26), (29) θα αντικατασταθούν από τις ακόλουθες εξισώσεις:

$$\nabla b_u Euit = -e_{uit} + \lambda b_u \quad (44)$$ (η οποία στην ουσία είναι η ίδια)

$$\nabla a_u Euit = -e_{uit} * dev_u(t) + \lambda a_u \quad (45)$$

$$\nabla b_{u,t} Euit = -e_{uit} + \lambda b_{u,t} \quad (46)$$

Η βελτιστοποίηση της συνάρτησης σφάλματος λαμβάνοντας υπόψη τις χρονικές παραμέτρους του μοντέλου θα εκτελεστή με τη εφαρμογή του stochastic Gradient Descent αλγορίθμου.

5.2. Περίληψη αλγόριθμου

- Επανέλαβε μέχρι να συγκλίνει:
  - Για κάθε (u,i,t) ∈ K:
    - Υπολόγισε το σφάλμα: $e_{uit} \leftarrow r_{ui}(t) - \hat{r}_{ui}(t)$
    - Ενημέρωσε: $X \leftarrow X - \gamma \nabla_X Euit$
      - ∀ $j \in P_{(u,t)}$:

Ενημέρωσε $y \leftarrow y - \gamma \nabla_y Euit$

- ∀ $j \in C_{(u,t)}$:

Ενημέρωσε $y' \leftarrow y' - \gamma \nabla_{y'} Euit$

- ∀ $j \in N_{(u)}^{catch}$:

Ενημέρωσε $\mathcal{Z} \leftarrow \mathcal{Z}$-$\gamma \nabla_{\mathcal{Z}}$Euit

- ∀ $j \in N_{(u)}^{VOD}$:

Ενημέρωσε $\mathcal{Z}' \leftarrow \mathcal{Z}'$-$\gamma \nabla_{\mathcal{Z}'}$Euit

5.2.1. Stochastic Gradient Descent Algorithm

Επανέλαβε μέχρι να συγκλίνει:
Για κάθε (u,i,t) ∈ K [ΜΟΝΤΕΛΟ Ι]:

- Υπολόγισε: $e_{uit} \leftarrow r_{ui}(t) - \hat{r}_{ui}(t)$ (τρέχον σφάλμα πρόβλεψης) (47)
- Ενημέρωσε: $b_u^0 \leftarrow b_u^0 + \gamma\,(e_{uit} - \lambda b_u^0)$ (48)
- Ενημέρωσε:

$\underline{b}^r \leftarrow \underline{b}^r + \gamma\,(e_{uit} * [\Delta t * prev_u(t) + dev_u(t) * curr_u(t)] - \lambda * [(prev_u(t) + curr_u(t)) \odot \underline{b}^r])$ (49)

- Ενημέρωσε: $b_{u,t} \leftarrow b_{u,t} + \gamma\,(e_{uit} - \lambda b_{u,t})$ (50)
- Ενημέρωσε: $b_i \leftarrow b_i + \gamma\,(e_{uit} - \lambda b_i)$ (51)
- Ενημέρωσε: $b_{i,BIN(t)} \leftarrow b_{i,BIN(t)} + \gamma\,(e_{uit} - \lambda b_{i,BIN(t)})$ (52)
- Ενημέρωσε: $b_{i,DAY(t)} \leftarrow b_{i,DAY(t)} + \gamma\,(e_{uit} - \lambda b_{i,DAY(t)})$ (53)
- Ενημέρωσε: $\underline{P}_u^0 \leftarrow \underline{P}_u^0 + \gamma\,(e_{uit} * q_i - \lambda \underline{P}_u^0)$ (54)
- Ενημέρωσε: $\underline{P}_{u,t} \leftarrow \underline{P}_{u,t} + \gamma\,(e_{uit} * q_i - \lambda \underline{P}_{u,t})$ (55)
- ∀ $j \in P_{(u,t)}$ (σύνολο των προηγούμενων χρονικών διαστημάτων):

Ενημέρωσε: $\underline{P}_u^j \leftarrow \underline{P}_u^j + \gamma\,(e_{uit} * \Delta t * \underline{q}_i - \lambda \underline{P}_u^j)$ (56)

- ∀ $j \in C_{(u,t)}$ (συνολο δεικτών του τρέχοντος χρόνου):

Ενημέρωσε: $\underline{P}_u^j \leftarrow \underline{P}_u^j + \gamma\,(e_{uit} * dev_u(t) * \underline{q}_i - \lambda \underline{P}_u^j)$ (57)

- ∀ $j \in N_{(u)}^{catch}$ (σύνολο δεικτών των στοιχείων catch up)

Ενημέρωσε: $Y_j^{catch} \leftarrow Y_j^{catch} + \gamma(|N_{(u)}^{catch}|^{-\frac{1}{2}} * e_{uit} * \underline{q}_i - \lambda Y_j^{catch})$ (58)

- ∀ $j \in N_{(u)}^{VOD}$: (σύνολο δεικτών των στοιχείων vod)

Ενημέρωσε: $Y_j^{VOD} \leftarrow Y_j^{VOD} + \gamma(|N_{(u)}^{VOD}|^{-\frac{1}{2}} * e_{uit} * q_i - \lambda Y_j^{VOD})$ (59)

Στην περίπτωση που το αρχικό γραμμικό μοντέλο του βασικού εκτιμητή χρήστη χρησιμοποιηθεί, οι εξισώσεις (48), (49) πρέπει να αντικατασταθούν από [ΜΟΝΤΕΛΟ ΙΙ]:

Ενημέρωσε: $b_u \leftarrow b_u + \gamma\,(e_{uit} - \lambda b_u)$ (60)

Ενημέρωσε: $a_u \leftarrow a_u + \gamma\,(e_{uit} * dev_u(t) - \lambda a_u)$ (61)

Προσοχή το $dev_u(t)$ είναι τώρα διαφορετική ποσότητα

5.2.2. Αλγοριθμική παράγοντες

1. Προσδιορίζεται ότι το χρονικό διάστημα των αλληλεπιδράσεων για κάθε χρήστη: $\{t^u_{min}, t^u_{max}\}$, $\forall$ u∈U
2. Προσδιορισμός του αριθμό των ενδιάμεσων σημείων ελέγχου κατά μήκος του άξονα του χρόνου αλληλεπιδράσεων του κάθε χρήστη K_u. Ένας αριθμός (K_u) των ενδιάμεσων σημείων ελέγχου προκαλεί έναν συνολικό αριθμό (K_u + 2) χρονικά σημεία κατά μήκος του άξονα χρόνου (συμπεριλαμβανομένου του start / stop σημείου $\{t^u_{min}, t^u_{max}\}$) και ένας αριθμός (K_u + 1) τοποθετούνται σε ίσα χρονικά διαστήματα. Ο αριθμός των επαγόμενων χρονικά διαστημάτων είναι επίσης ίσο με τον αριθμό των $\underline{P}^j_u$ διανυσμάτων που χρησιμοποιούνται για τη μοντελοποίηση του χρόνου διακύμανση των διανυσμάτων χρήστη και του διανύσματος $\underline{b}^v$. Ως εκ τούτου, ισχύει ότι:

   $\forall$ (u,i,t)∈ K : $|P_{(u,t)}|+|C_{(u,t)}| \leq K_u + 1$

   (Δεδομένου ότι $|C_{(u,t)}|=1$, τότε έχουμε $|P_{(u,t)}| \leq K_u$)

3. Καθορισμός του μέσου σημείου για κάθε χρήστη: $t_u = \frac{t^u_{max} - t^u_{min}}{2}$ στην περίπτωση που το αρχικό γραμμικό μοντέλο χρησιμοποιείται
4. Καθορισμός του πλήθους των διαστημάτων στο οποίο θα διαμεριστεί ο συνολικός άξονας του χρόνου για όλα τα αντικείμενα, μέσω του προσδιορισμού της ελάχιστης και της μέγιστης χρονικά στιγμής για κάθε βαθμολογία.
5. Καθορισμός των παραμέτρων που λαμβάνονται υπόψη στο συνολικό μοντέλο
6. Αρχικοποίηση (τυχαία) σε κάθε μια από τις εσωτερικές παραμέτρους.
7. Εφαρμογή αλγόριθμων (MODEL I) και (MODEL II) για την ελαχιστοποίηση της συνάρτησης λάθους σε ένα σύνολο εκπαίδευση και ένα σύνολο εκτίμησης και δοκιμή RMSE.
8. Πριν εφαρμοστεί το βήμα 7 ένα πλάνο κατάτμησης του συνολικού των σημείων χρειάζεται προκειμένου να λάβουμε πιο ακριβή αποτελέσματα εκτίμησης.

5.2.3. Λίστα παραμέτρων

[P1]: $b^0_u \in \mathbb{R}$ (μια για κάθε χρήστη u ∈ U)

[P2]: $\underline{b}^{U} \in \mathbb{R}^{K_u+1}$ (ένα διάνυσμα για κάθε χρήστη u ∈ U)

[P3]: $b_{u,t} \in \mathbb{R}$ (μια για κάθε χρήστη και αντίστοιχο στιγμιότυπο χρόνου)

[P4]: $b_i \in \mathbb{R}$ (μια για κάθε αντικείμενο i ∈ I)

[P5]: $b_{i,\,BIN}(t)$ (μια για κάθε αντικείμενο και αντίστοιχο διάστημα χρόνου στον άξονα χρόνου όλων των αντικειμένων)

[P6]: $b_{i,\,DAY}(t)$ (μια για κάθε στοιχείο και για την αντιστοιχεί χρονική στιγμή της ημέρας)

[P7]: $\underline{P}_u^0 \in \mathbb{R}^f$ (ένα διάνυσμα για κάθε χρήστη u ∈ U)

[P8]: $\underline{P}_{u,t} \in \mathbb{R}^f$ (ένα διάνυσμα για κάθε χρήστη και την αντίστοιχη χρονική στιγμή)

[P9]: $\underline{P}_u^j \in \mathbb{R}^f$ (ένα διάνυσμα για κάθε χρήστη και τον αντίστοιχο δείκτη χρονικού διαστήματος (παρελθόν))

[P10]: $\underline{P}_u^j \in \mathbb{R}^f$ (ένα διάνυσμα για κάθε χρήστη και το αντίστοιχο (τρέχον) χρονικό διάστημα)

[P11]: $Y_j^{catch} \in \mathbb{R}^f$ (μια για κάθε στοιχείο catch up)

[P12]: $Y_j^{VOD} \in \mathbb{R}^f$ (μια για κάθε στοιχείο vod)

[P13]: $b_u \in \mathbb{R}$ (μια για κάθε χρήστη u ∈ U) [*η P13 είναι ίδια με την P1]

[P14]: $a_u \in \mathbb{R}$ (μια για κάθε χρήστη one u ∈ U)

Σημειώνεται και πάλι ότι οι παράμετροι [P1], [P2] και [P13], [P14] είναι συμπληρωματικοί.

6. Συμπεράσματα & Μελλοντικές Ενέργειες

Η πειραματική διαδικασία για τον έλεγχο της ακρίβεια του προτεινόμενου αλγορίθμου παρουσιάζεται παρακάτω.

Το πλήρες σύνολο δεδομένων διαμερίστηκε με τυχαίο τρόπο σε 57 υποσύνολα των 1000 χρηστών. Συγκεκριμένα το πρώτο υποσύνολο περιελάμβανε 1000 χρήστες και 15.535 αντικείμενα προγράμματος. Για το συγκεκριμένο υποσύνολο δεδομένων ήταν διαθέσιμες 1.495.727 τιμές βαθμολόγησης που αντιστοιχούν σε sparsity = 1495727/1000*15535=0,0963. Για το συγκεκριμένο πρόβλημα το πλήθος των διαθέσιμων αξιολογήσεων διαμερίστηκε με τυχαίο τρόπο χρησιμοποιώντας το 80% για training και 20% για testing. Η ταξινομητική ακρίβεια κατά την διάρκεια της εκπαίδευσης ήταν 1,1436 ενώ κατά τη διάρκεια του ελέγχου ήταν 1,1572.

Το δεύτερο υποσύνολο περιελάμβανε 1000 χρήστες και 15.554 αντικείμενα προγράμματος. Για το συγκεκριμένο υποσύνολο δεδομένων ήταν διαθέσιμες 1.578.688 τιμές βαθμολόγησης που αντιστοιχούν σε sparsity =0,1015. Για το συγκεκριμένο πρόβλημα το πλήθος των διαθέσιμων αξιολογήσεων διαμερίστηκε με τυχαίο τρόπο χρησιμοποιώντας το 80% για training και 20% για testing. Η ταξινομητική ακρίβεια κατά την διάρκεια της εκπαίδευσης ήταν 1,1698 ενώ κατά τη διάρκεια του ελέγχου ήταν 1,1826.

Παρακάτω παρουσιάζεται ο πίνακας με όλα τα πειράματα που πραγματοποιήθηκαν.

Α/Α πειράματος	Αρ. Χρηστών	Αντικείμενα	Διαθέσιμές αξιολογήσεις	Sparsity	Mae Train	Mae Test
1	1000	15.535	1.495.727	0,0963	1,1436	1,1572
2	1000	15.554	1.578.688	0,1015	1,1698	1,1826
3	1000	15.530	1.579.175	0,1017	1,2023	1,2114
4	1000	15.517	1.474.633	0,0950	1,3194	1,3323
5	1000	15.550	1.522.013	0,0979	1,2500	1,2510
6	1000	15.568	1.528.249	0,0982	1,1799	1,1900
7	1000	15.552	1.527.791	0,0982	1,1938	1,2050
8	1000	15.560	1.509.797	0,0970	1,2592	1,2693
9	1000	15.554	1.569.589	0,1009	1,1692	1,1825
10	1000	15.565	1.506.780	0,0968	1,1912	1,1997
11	1000	15.548	1.525.358	0,0981	1,1744	1,1844

12	1000	15.553	1.525.997	0,0981	1,1714	1,1816
13	1000	15.533	1.412.923	0,0910	1,2916	1,2995
14	1000	15.554	1.508.024	0,0970	1,1845	1,1951
15	1000	15.550	1.578.593	0,1015	1,2389	1,2422
16	1000	15.536	1.507.474	0,0970	1,1832	1,1897
17	1000	15.542	1.500.452	0,0965	1,1952	1,2050
18	1000	15.549	1.546.007	0,0994	1,1902	1,2068
19	1000	15.543	1.553.075	0,0999	1,1842	1,1952
20	1000	15.536	1.510.161	0,0972	1,1895	1,2010
21	1000	15.542	1.422.480	0,0915	1,2759	1,2886
22	1000	15.530	1.561.346	0,1005	1,1521	1,1533
23	1000	15.536	1.506.624	0,0970	1,2490	1,2531
24	1000	15.553	1.544.242	0,0993	1,1552	1,1567
25	1000	15.535	1.544.037	0,0994	1,1864	1,1872
26	1000	15.553	1.538.270	0,0989	1,1908	1,2078
27	1000	15.547	1.521.228	0,0978	1,1619	1,1645
28	1000	15.558	1.579.525	0,1015	1,1927	1,2029
29	1000	15.547	1.563.824	0,1006	1,1889	1,1995
30	1000	15.539	1.550.034	0,0998	1,1895	1,1965
31	1000	15.539	1.535.616	0,0988	1,1864	1,2010
32	1000	15.564	1.520.805	0,0977	1,2666	1,2740
33	1000	15.540	1.538.283	0,0990	1,1756	1,1861
34	1000	15.539	1.498.875	0,0965	1,1702	1,1861
35	1000	15.555	1.508.667	0,0970	1,1878	1,2019
36	1000	15.545	1.507.822	0,0970	1,1717	1,1823

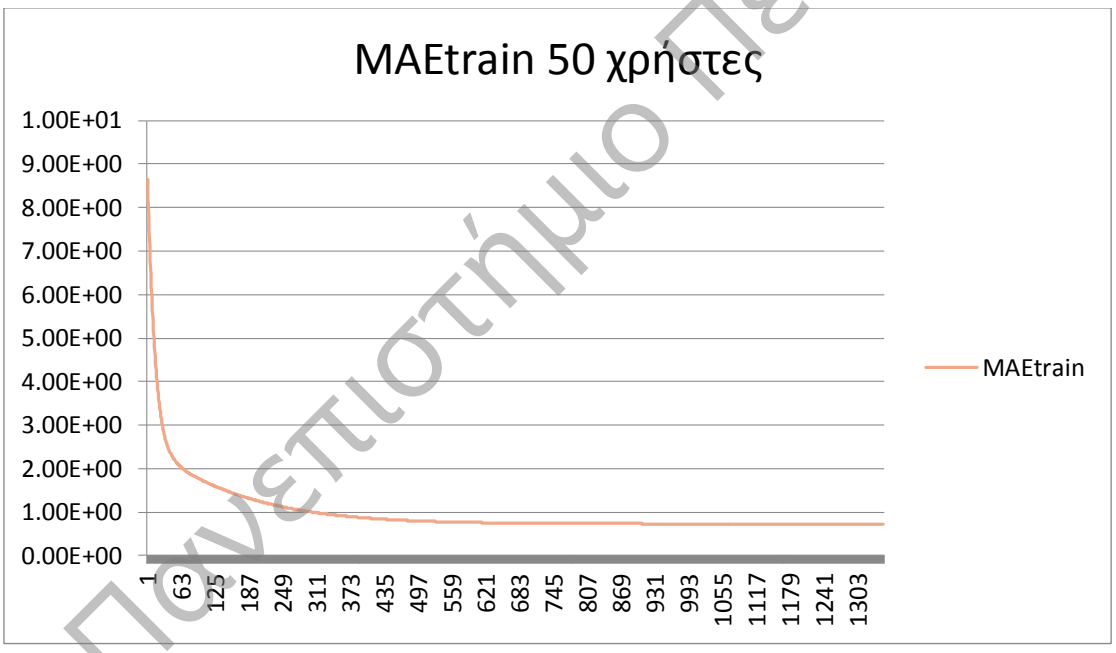
37	1000	15.536	1.446.842	0,0931	1,1864	1,1972
38	1000	15.555	1.514.566	0,0974	1,2521	1,2592
39	1000	15.567	1.516.740	0,0974	1,1879	1,1884
40	1000	15.539	1.522.656	0,0980	1,1609	1,1694
41	1000	15.561	1.552.685	0,0998	1,1956	1,2057
42	1000	15.546	1.575.605	0,1014	1,1912	1,2017
43	1000	15.546	1.446.123	0,0930	1,1831	1,1893
44	1000	15.554	1.499.335	0,0964	1,1871	1,2010
45	1000	15.563	1.542.418	0,0991	1,1593	1,1800
46	1000	15.536	1.572.235	0,1012	1,1543	1,1641
47	1000	15.540	1.511.690	0,0973	1,1521	1,1810
48	1000	15.569	1.547.443	0,0994	1,2227	1,2807
49	1000	15.543	1.558.990	0,1003	1,1540	1,1596
50	1000	15.531	1.540.656	0,0992	1,1728	1,1930
51	1000	15.572	1.563.781	0,1004	1,1961	1,2020
52	1000	15.564	1.453.761	0,0934	1,1757	1,1966
53	1000	15.548	1.558.042	0,1002	1,1860	1,2019
54	1000	15.561	1.577.195	0,1014	1,1971	1,1975
55	1000	15.563	1.526.411	0,0981	1,2485	1,2575
56	1000	15.577	1.513.708	0,0972	1,1987	1,1992
57	1000	15.546	1.529.970	0,0984	1,1916	1,1926

Η παράμετρος γ του αλγορίθμου ήταν 10^{-6} και λ=1. Για τα διανύσματα των χρηστών και των αντικειμένων που εκτιμήθηκαν η διάσταση των χαρακτηριστικών τους ήταν ίση με 10.

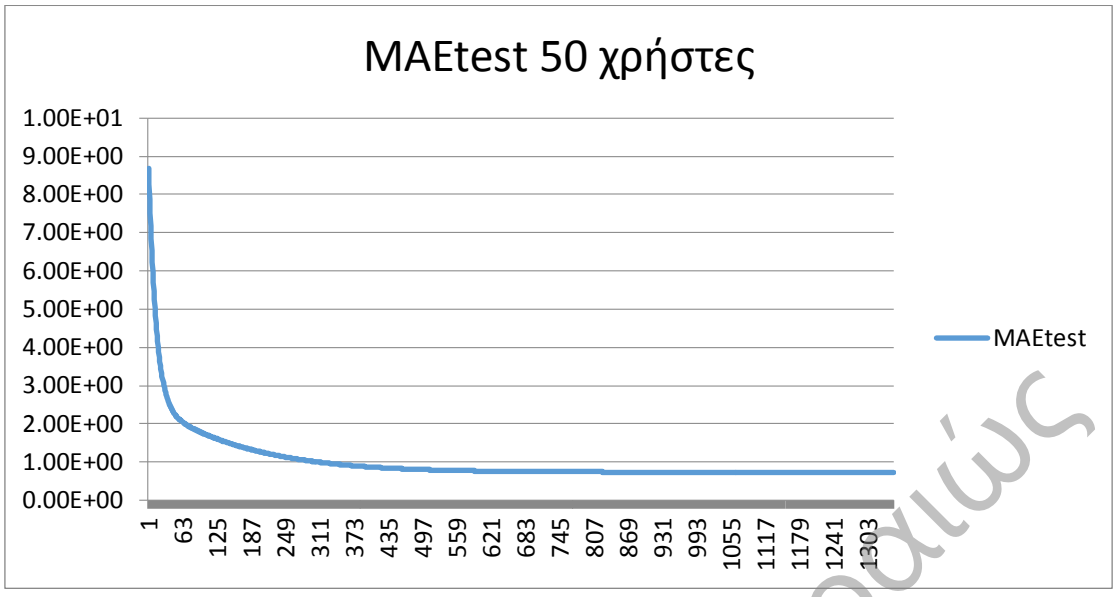
Το επόμενο πείραμα που πραγματοποιήθηκε ήταν από το σύνολο των χρηστών να επιλεγούν οι 50, 100, 250, 500 και 1000 χρήστες με τις περισσότερες αξιολογήσεις τα αποτελέσματα παρουσιάζονται παρακάτω:

Α/Α πειράματος	Αρ. Χρηστών	Αντικείμενα	Διαθέσιμές αξιολογήσεις	Sparsity	Mae Train	Mae Test
1	50	15.224	272.674	0,358	0,7308	0,7345
2	100	15.543	531.070	0.342	0,7502	0,7526
3	250	15.709	1.235.982	0,314	0,8238	0,8249
4	500	15.758	2.346.688	0,298	0,8450	0,8476
5	1000	15.780	4.360.718	0,276	0,8576	0,8578

Για το πείραμα των 50 χρηστών σχεδιάστηκαν οι καμπύλες μάθησης και παρουσιάζονται παρακάτω.



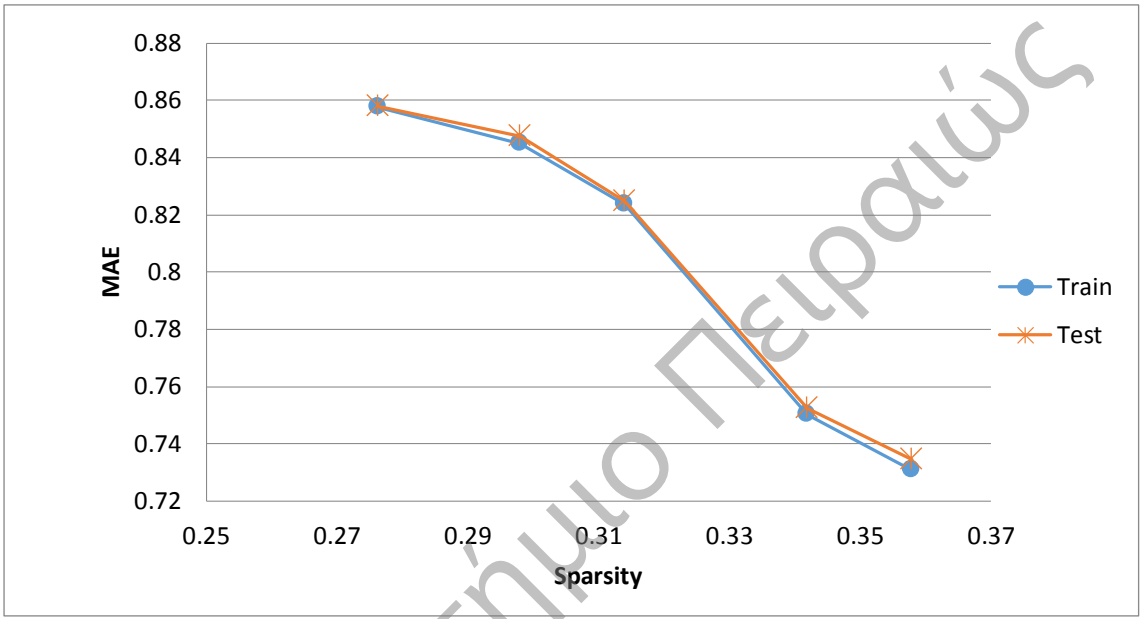
Εικόνα 7: Καμπύλη εκπαίδευσης MAEtrain



Εικόνα 8: Καμπύλη ελέγχου MAEtest

Η διαδικασία ελέγχου της εγκυρότητας του προτεινόμενου αλγορίθμου κατέδειξε πρωτίστως την αποτελεσματικότητά του για το πρόβλημα της σύστασης πολυμεσικών αντικειμένων μέσα στο ευρύτερο πλαίσιο της εξατομίκευσης των υπηρεσιών IPTV. Συγκεκριμένα, τα πειραματικά αποτελέσματα που συλλέχθηκαν αφορούν σε ένα σύνολο δοκιμών σε διαφορετικά υποσύνολα χρηστών και αντικειμένων με διαφορετικά επίπεδα αραιότητας δεδομένων (sparsity). Τα εν λόγω αποτελέσματα υποδεικνύουν σημαντική βελτίωση της τιμής του μέσου απόλυτου σφάλματος (MAE) τόσο κατά την διαδικασία της εκπαίδευσης όσο και κατά την διαδικασία του ελέγχου σε σχέση με την αντίστοιχη μέγιστη τιμή του που θα προέκυπτε από έναν εκφυλισμένο αλγόριθμο σύστασης βασιζόμενο στην καθαρή τύχη. Επιπλέον οι τιμές του μέσου απόλυτου σφάλματος κατά την διαδικασία της εκπαίδευσης και κατά την διαδικασία του ελέγχου για το σύνολο των πειραμάτων που διενεργήθηκαν ταυτίζονται με ακρίβεια πρώτου δεκαδικού ψηφίου αποδεικνύοντας την ικανότητα γενίκευσης και την σταθερότητα του προτεινόμενου μοντέλου. Μάλιστα, η βελτίωση σε σχέση με την μέγιστη τιμή του μέσου απόλυτου σφάλματος (MAX MAE = 2.5) για τα καλύτερα αποτελέσματα που συλλέχθηκαν ήταν της τάξης του 1.75 στην αντίστοιχη κλίμακα του MAE για τιμές rating στο διακριτό διάστημα {1,2,3,4,5}.

Ένα σημαντικό συμπέρασμα που προκύπτει από τον πειραματικό έλεγχο του αλγορίθμου σύστασης έχει να κάνει με την σχέση μεταξύ της ακρίβειας σύστασης και της πυκνότητας των διαθέσιμων αξιολογήσεων για κάθε υποσύνολο δεδομένων που χρησιμοποιήθηκε κατά την διαδικασία του πειραματισμού. Συγκεκριμένα, πραγματοποιήθηκε μια ακολουθία πειραμάτων με αυξανόμενο πλήθος χρηστών και διαθέσιμων τιμών rating που αντιστοιχούσε όμως σε μειούμενο βαθμό αραιότητας δεδομένων. Τα αποτελέσματα που προέκυψαν από αυτή την διαδικασία υποδεικνύουν σχετική αυξητική τάση στις τιμές του μέσου απόλυτου σφάλματος καθώς μειώνονται οι τιμές της αραιότητας των δεδομένων, καθιστώντας δυσκολότερο το αντίστοιχο πρόβλημα ταξινόμησης.



Εικόνα 9: Σχέση ταξινομικής ακρίβειας και sparsity

Ωστόσο, η ταξινομητική ακρίβεια του αλγορίθμου εξακολουθεί να κινείται σε ικανοποιητικά επίπεδα ακόμα για ακραία μικρές τιμές αραιότητας δεδομένων. Μάλιστα, αξίζει να σημειωθεί πως για ένα εύρος τιμών αραιότητας στο διάστημα 0.09 έως 0.35 οι αντίστοιχες τιμές του μέσου απόλυτου σφάλματος κινούνται στο διάστημα 1.2 έως 0.75, αποδεικνύοντας την σταθερότητα του αλγορίθμου σε ακραίες συνθήκες έλλειψης δεδομένων.

Ολοκληρώνοντας την ανάλυση των συμπερασμάτων σχετικά με την συμπεριφορά του προτεινόμενου αλγορίθμου θα πρέπει να αναφερθεί η ανεκτικότητά του στις έντονες συνθήκες ταξικής ανισορροπίας που διέπει την κατανομή των τιμών του rating για το σύνολο των διαθέσιμων χρηστών σε κάθε υποσύνολο δεδομένων που χρησιμοποιήθηκε κατά τον πειραματικό έλεγχο του αλγορίθμου. Συγκεκριμένα, λαμβάνοντας υπόψιν το γεγονός πως η κατανομή των τιμών του rating για το αντίστοιχο κάθε φορά σύνολο δεδομένων που χρησιμοποιήθηκαν για εκπαίδευση συγκεντρώνει το μεγαλύτερο ποσοστό της μάζας της στην τιμή 1, θα ανέμενε κανείς μια παρόμοια συμπεριφορά για τις εκτιμώμενες τιμές rating που προκύπτουν από την δράση του αλγορίθμου. Ωστόσο, μια τέτοιου είδους συμπεριφορά δεν παρατηρήθηκε με τις εκτιμώμενες τιμές του rating να κατανέμονται κανονικά γύρω από την μέση τιμή του rating για το σύνολο των δεδομένων εκπαίδευσης.

Στις μελλοντικές επεκτάσεις της παρούσας εργασίας συγκαταλέγεται η σύγκριση του προτεινόμενου αλγορίθμου με άλλα μοντέλα σύστασης με χρόνο-επίγνωση που υπάρχουν στην τρέχουσα βιβλιογραφία των συστημάτων σύστασης. Η συγκεκριμένη πρακτική κρίνεται ιδιαίτερα σημαντική καθώς η σωστή μοντελοποίηση της χρονικής μεταβολής των προτιμήσεων των χρηστών είναι το βασικότερο χαρακτηριστικό σε ένα σύστημα σύστασης που αφορά υπηρεσίες IPTV. Η διενέργεια περισσότερων πειραμάτων και η προσπάθεια ταυτόχρονης προγραμματιστικής διαχείρισης μεγαλύτερου όγκου δεδομένων αποτελούν προκλήσεις που καθιστούν αναγκαία την μελλοντική παραλληλοποίηση του αλγορίθμου. Η επίλυση των εν λόγω προβλημάτων μπορεί να οδηγήσει στην υλοποίηση εξατομικευμένων συστημάτων σύστασης για τα οποία υπάρχει το συγκριτικό πλεονέκτημα που σχετίζεται με την μη αναγκαιότητα ύπαρξης ρητών αξιολογήσεων από την πλευρά του χρήστη. Συνοψίζοντας, μια από τις πιο αξιόλογες ερευνητικά δυνατότητες επέκτασης της παρούσας εργασίας είναι η διατύπωση μοντέλων και αντίστοιχων αλγορίθμων για την μοντελοποίηση ομάδων χρηστών (group modelling).

7. Βιβλιογραφία

1. Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35
2. Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, The VITA Financial Services Sales Support Environment, in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007.
3. Christakis Nicholas A. and Fowler James H., Connected: The Surprising Power of Our Social Networks and How They Shape Our Lives. Little, Brown and Co. 2009.
4. Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: a Survey of the State-of-the-art and Possible Extensions. IEEE Transactions on Knowledge Management and Data Engineering, 17(6): 734-749
5. www.ibm.com | TCP/IP Tutorial and Technical Overview
6. www.nearearthllc.com | IPTV – The Future of Television
7. http://el.wikipedia.org/wiki/Τηλεόραση
8. J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Proc. 14th Conf. Uncertainty in Artificial Intelligence, July 1998.
9. C.C. Aggarwal, J.L. Wolf, K-L. Wu, and P.S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering," Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, Aug. 1999.
10. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," Proc. 10th Int'l WWW Conf., 2001.
11. M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation Algorithms," ACM Trans. Information Systems, vol. 22, no. 1, pp. 143-177, 2004.
12. Yehuda Koren and Robert Bell, "Advances in Collaborative Filtering", Recommender Systems Handbook, Springer, 2011, pp. 145-186
13. Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model", Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.
14. Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model", Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.
15. Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model", Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008.

ΠΑΡΑΡΤΗΜΑ Ι Ονόματα μεταβλητών

1. *UsersNumber*: |U|, ο αριθμός των μοναδικών χρηστών
2. *ItemsNumber*: |I|, ο αριθμός των μοναδικών αντικειμένων.
3. TimeIntervalsNumber: Ένα [1x UsersNumber] διάνυσμα που αποθηκεύει τον αριθμό των χρονικών διαστημάτων που λαμβάνεται υπόψη για κάθε χρήστη.
4. ControlPointsNumber: ένα [1x UsersNumber] διάνυσμα αποθήκευσης του αριθμού των χρονικών διαστημάτων που πρέπει να εξεταστεί για κάθε χρήστη.

   [Η μεταβλητή TimeIntervalsNumber χρειάζεται να οριστεί πριν την μεταβλητή ControlPointsNumber αφού ισχύει ότι: ControlPointsNumber(u)= TimeIntervalsNumber(u)+1]

5. StartingPoints: ένα [1xUsersNumber] διάνυσμα που αποθηκεύει την χρονική στιγμή της έναρξης αλληλεπίδρασης ενός συγκεκριμένου χρήστη.
6. EndingPoints: ένα [1xUsersNumber] διάνυσμα που αποθηκεύει την χρονική στιγμή του τερματισμού της αλληλεπίδρασης ενός συγκεκριμένου χρήστη.
7. ControlPoints: ένας (numpy) πίνακας από γραμμές UsersNumber που αποθηκεύει τα σημεία ελέγχου κατά μήκος του άξονα χρόνου των αλληλεπιδράσεων για κάθε χρήστη. Σημειώνεται ότι κάθε φορέας περιλαμβάνει το σημεία έναρξης και τέλους καθώς και ότι όλα τα διανύσματα δεν είναι απαραίτητα το ίδιο μήκος.
8. UsersRatingNumber: Ένα [1xUsersNumber] διάνυσμα που αποθηκεύει τον αριθμό των αξιολογήσεων ενός συγκεκριμένου χρήστη. Αυτός ο αριθμός είναι προφανώς ίσος με τον αριθμό των χρονικών διαστημάτων αλληλεπίδρασης ενός συγκεκριμένου χρήστη. για παράδειγμα τις αλληλεπιδράσεις του συγκεκριμένου χρήστη.
9. FeaturesDimensionality: Η διάσταση διανυσμάτων των χρηστών και των στοιχείων που πρέπει να θεωρηθεί (εσωτερική παράμετρος f).
10. UsersBaselineInitialTerm: Ένα [1xUsersNumber] διάνυσμα που αποθηκεύει την αρχική διάρκεια του βασικού εκτιμητή για κάθε χρήστη.
11. UsersBaselineTimeInternalTerms: Ένας (numpy) πίνακας UsersNumber γραμμών που αποθηκεύει σε κάθε γραμμή τα βάρη (εσωτερικοί παράμετροι διανυσμάτων $\underline{b}^U$) που συνδέονται με κάθε χρονικό διάστημα. Η u-στη γραμμή έχει ένα αριθμό από TimeIntervalsNumber(u) στοιχεία.
12. UsersBaselineTimeInstanceTerm: Ένας (numpy) πίνακας από UsersNumber γραμμές που αποθηκεύουν σε κάθε γραμμή τους όρους που αντιστοιχούν στα χρονικά στιγμιότυπα των βασικών εκτιμητών του χρήστη (εσωτερικές παράμετροι $b_{u,t}$). Η u-στη γραμμή έχει έναν αριθμό από UsersRatingsNumber(u) στοιχεία.
13. ItemsBaselineInitialTerm: Ένα [1xItemsNumber] διάνυσμα που αποθηκεύει τους αρχικούς όρους του βασικού εκτιμητή για κάθε αντικείμενο.

14. BinsNumber: Το πλήθος των διαστημάτων που θα πρέπει να διαμεριστεί ο άξονας του χρόνου για το σύνολο των αντικειμένων. Η επιλογή της παραμέτρου εξαρτάται από το συνολικό χρονικό διάστημα στο οποίο κινούνται οι βαθμολογίες. Αυτή η παράμετρος είναι συγκεκριμένο στοιχείο και θα μπορούσε να είναι ο αριθμός των μηνών που καλύπτουν το πλήρες χρονοδιάγραμμα ή κάποιο προκαθορισμένο αριθμό εβδομάδων.
15. DaysNumber: Ο αριθμός των ημερών μέσα σε μια εβδομάδα.
16. UsersVectorsInitialTerm: Ένας (numpy) πίνακας μεγέθους [UsersNumber x FeaturesDimensionality]. Αυτός ο πίνακας έχει ακριβώς UsersNumber γραμμές από FeaturesDimensionality στοιχείων. Κάθε σειρά αποθηκεύει την αρχική διάρκεια του διανύσματος για το διάνυσμα της γνωστής ώρας του χρήστη (εσωτερική παράμετρος $\underline{P}_u^0$).

$$\underline{\underline{\mathbb{P}}}^0 = \begin{bmatrix} \underbrace{\qquad\qquad}_{f} : P_1^0 \\ \underbrace{\qquad\qquad}_{f} : P_2^0 \\ \vdots \\ \underbrace{\qquad\qquad}_{f} : P_{|U|}^0 \end{bmatrix} \text{ (20 array)}$$

17. UsersVectorTimeIntervalTerm: Είναι μια λίστα της python από UsersNumber στοιχεία. Το u-στο στοιχείο αυτής της λίστας είναι ένας συγκεκριμένος (numpy) πίνακας μεγέθους [TimeIntervalsNumber(u)x FeaturesDimensionality]. Κάθε πίνακας στη λίστα αποθηκεύει K_u+1 = TimeIntervalsNumber(u) διανύσματα από FeaturesDimensionality στοιχεία. Αυτά τα διανύσματα αντιστοιχούν στην εσωτερική παράμετρο $\underline{P}_u^j \in \mathbb{R}^f$).

$$\underline{\underline{\mathbb{P}}}^{interval} = \begin{bmatrix} \begin{bmatrix} \underbrace{\qquad\qquad}_{f} : P_1^1 \\ \underbrace{\qquad\qquad}_{f} : P_1^2 \\ \vdots \\ \underbrace{\qquad\qquad}_{f} : P_1^{K_1+1} \end{bmatrix} \cdots \begin{bmatrix} \underbrace{\qquad\qquad}_{f} : P_{|U|}^1 \\ \underbrace{\qquad\qquad}_{f} : P_{|U|}^2 \\ \vdots \\ \underbrace{\qquad\qquad}_{f} : P_{|U|}^{K_{|U|}+1} \end{bmatrix} \end{bmatrix} \text{ (3D array)}$$

$$\mathbb{P}^{timeinstance} = \left[\left[\underbrace{\square}_{f}: P_{1,1} \;\; \underbrace{\square}_{f}: P_{1,2} \;\; \vdots \;\; \underbrace{\square}_{f}: P_{1,|K_{(1)}|}\right] \cdots \left[\underbrace{\square}_{f}: P_{|U|,1} \;\; \underbrace{\square}_{f}: P_{|U|,2} \;\; \vdots \;\; \underbrace{\square}_{f}: P_{|U|,|K_{(|U|)}|}\right]\right]$$

- Επεξήγηση των ακριβών διαστάσεων των μεταβλητών 18, 19, 20 που προέρχονται από το αρχικό ορισμό της γνώσης του χρόνου των διανυσμάτων χρήστη που δίνεται από την εξίσωση

$$\underline{P}_u(t) = \underbrace{\underbrace{\underline{P}_u^0}_{\text{initial term}} + \underbrace{\sum_{j\epsilon P(u,t)} \Delta t * \underline{P}_u^j + \sum_{j\epsilon C(u,t)} dev_u(t) * \underline{P}_u^j}_{\text{time-interval specific terms}}}_{} + \underbrace{\underline{P}_{u,t}}_{\text{time-instance specific terms}}$$

$\underline{P}_u^0 \in \mathbb{R}^f$: ένα ανά χρήστη

$\underline{P}_u^j \in \mathbb{R}^f$: (K_u+1) ή TimeIntervalsNumber(u) ανά χρήστη από

$|P(u,t)|+|C(u,t)| = K_u+1 =$ TimeIntervalsNumber(u)

18. UsersCatchupNumber: Ένα [1x UsersNumber] διάνυσμα που αποθηκεύει τον αριθμό των στοιχείων catchup ανά χρήστη.
19. UsersVodNumber: Ένα [1x UsersNumber] διάνυσμα που αποθηκεύει τον αριθμός των vod στοιχείων ανά χρήστη.
20. CatchupNumber: Ο αριθμός των στοιχείων catchup.
21. VodNumber: Ο αριθμός των στοιχείων vod.
22. UsersCatchupIndires: μια λίστα της python που περιέχει στοιχεία UsersNumber. Κάθε (u) στοιχείο είναι ένας πίνακας NumPy διανυσμάτων UsersCatchupNumber(u) τιμών που αποθηκεύει τους δείκτες των αντικειμένων που έχουν γίνει catched up από τον χρήστη (u).
23. UsersVodIndires: Μια λίστα της python που περιέχει στοιχεία UsersNumber. Το u στοιχείο είναι ένας πίνακας (NumPy) διανυσμάτων με UsersVodNumber(u) που οι τιμές που αποθηκεύει είναι δείκτες από τα στοιχεία vod που αγοράστηκαν από τον χρήστη (u).

- CODING SAMPLES:
  Ένας πίνακας numpy μπορεί να αποθηκεύσει 220.000.000 τιμές και να εκτελέσει την λειτουργία αναζήτησης σε αυτόν. Για παράδειγμα μπορεί να γραφεί το παρακάτω:
  Εισαγωγή numpy μέχρι

X=np.random.randint(1,10,size=(220.000.000,1))
#Βρες όλες τις 1ες περιπτώσεις στο X
I1=np.where(x==1)
Ωστόσο, δεν μπορεί ταυτόχρονα να εκτελεστεί η αποθήκευση του X και 10 πινάκων τύπου I1, I2, ..., I10.I1, I2, …, I10

24. CatchupItemsVectors: Ένας πίνακας (numpy) με μέγεθος [CatchupNumber x FeaturesDimensionality] που αποθηκεύει τα διανύσματα στοιχείων για όλα τα στοιχεία που έχουν γίνει catched up

$$\text{CatchupItemsVectors} = \begin{bmatrix} \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_1^{catch} \\ \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_2^{catch} \\ \vdots \\ \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_{CatchupNumber}^{catch} \end{bmatrix}$$

25. VodItemsVectors: Ένας πίνακας (numpy) με μέγεθος [VodNumber x FeaturesDimensionality] που αποθηκεύει τα διανύσματα των στοιχείων για όλα τα vod στοιχεία.

$$\text{VodItemsVectors} = \begin{bmatrix} \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_1^{VOD} \\ \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_2^{VOD} \\ \vdots \\ \underbrace{\square\square\square\square\square\square\square\square\square\square}_{f} : Y_{VodNumber}^{VOD} \end{bmatrix}$$

26. UsersVectors: (numpy) [UsersNumber x FeaturesDimensionality].
27. ItemsVectors: (numpy) [ItemsNumber x FeaturesDimensionality].

<u>Ορισμοί βοηθητικών συνόλων</u>

U: Σύνολο μοναδικών χρηστών, |U| = 57826

I: σύνολο μοναδικών αντικειμένων, |I| = 16725

|K| ≈223.000.000

$K = \{(u,i,t) \in U \times I \times T : r_{ui}(t) \neq 0\}$ Αυτό είναι το σύνολο όλων των τριάδων (u,i,t) για τα οποία είναι γνωστή η βαθμολογία

$K(u^*)=\{(u,i,t)\in K: u=u^*\}$ Το σύνολο των τριάδων που αντιστοιχούν στην αξιολόγηση ενός συγκεκριμένου χρήστη

$K(i^*)=\{(u,i,t)\in K: i=i^*\}$ Το σύνολο όλων των τριάδων με γνωστές βαθμολογίες που αντιστοιχεί σε ένα συγκεκριμένο αντικείμενο.

$T(u^*)= \{t_1^{u^*}, t_2^{u^*}, \ldots, t_{|K(u^*)|}^{u^*}\} = \{t \in K: u=u^* \}$ Το σύνολο των χρονικών στιγμών όλων των γνωστών αξιολογήσεων του χρήστη u^*. Το σύνολο $T(u^*)$ διατάσσεται σε χρονολογική σειρά έτσι ώστε : $t_1^{u^*} = t_{min}^{u^*}$ και $t_{|K(u^*)|}^{u^*} = t_{max}^{u^*}$.

$T(i^*)= \{t_1^{i^*}, t_2^{i^*}, \ldots, t_{|K(i^*)|}^{i^*}\} = \{t \in K: i=i^* \}$ Το σύνολο των χρονικών διαστημάτων όλων των γνωστών αξιολογήσεων των αντικειμένων i^*. Το σύνολο $T(i^*)$ διατάσσεται σε χρονολογική σειρά έτσι ώστε : : $t_1^{i^*} = t_{min}^{i^*}$ και $t_{|K(i^*)|}^{i^*} = t_{max}^{i^*}$.

Ωστόσο, οι αντίστοιχες μεταβλητές που είναι εξαρτώμενες από τον χρόνο, δεν χρειάζεται να ταξινομηθούν σε χρονολογική σειρά

$$\text{DENSITY} = \frac{|U||I| - |K|}{|U||I|} \Rightarrow \text{DENSITY} \approx 76.94\%$$

ΠΑΡΑΡΤΗΜΑ ΙΙ Πηγαίος κώδικας

```
# This Python class manages all fundamental operations concerning the manipulation of the Tv data set.

# Our primary objective is to extend the original linear model for the users' baseline predictors proposed in

# [Handbook of Recommender Systems] by utilizing a partially linear model for each user. In this way we introduce

# a more flexible baseline predictor for each user in order to incorporate time-awareness within the recommendation process.

# Our aim is to compare the proposed model with the original linear one in terms of RMSE.

from __future__ import division

import os

import os.path as path

import logging

import pprint

import numpy as np

import scipy.io as spi

import math

import csv

import datetime

class  Recommender(object):

  def __init__(self,ratings_csv,catchup_csv,vod_csv,rated_items_csv):

    self.RatingsCsv = ratings_csv

    self.CatchupCsv = catchup_csv

    self.VodCsv = vod_csv

    print "Data source related parameters were successfully set"

  def quantize_rating_value(self,rating_value):

    if(0<=rating_value and rating_value<0.2):

    quantized_rating_value = 1
```

```
        elif(0.2<=rating_value and rating_value<0.4):
        quantized_rating_value = 2
        elif(0.4<=rating_value and rating_value<0.6):
        quantized_rating_value = 3
        elif(0.6<=rating_value and rating_value<0.8):
        quantized_rating_value = 4
        elif(0.8<=rating_value and rating_value<=1.0):
        quantized_rating_value = 5
        return quantized_rating_value
    def load_rated_items_ids(self):
        rated_items = []
        fp = open(self.RatedItemsCsv,'rb')
        reader = csv.reader( fp, delimiter=',', quotechar='"', escapechar='\\' )
        for row in reader:


        self.RatedItems = rated_items
        fp.close()
        # Do the printing only for testing reasons.
        #pprint.pprint(self.RatedItems)
    def
set_internal_parameters(self,ratings_number,columns_number,users_number,catchup_number,vod_number,catchup_records_number,vod_records_number):
        # INPORTANT!!!
        # All items within the final_zappings.csv file are identified by an integer number which is the id of the item.
        # However the series of ids is not an arithmetic sequence with unit step. Therefore, we cannot use these ids
        # as internal indices pointing to the corresponding variables. For this reason we need to construct a map, that is
        # a dictionary in python so that will be the link between a given item id and the real internal item index utilized
```

```
        # by the program. In this setting, ItemIndex[item_id] will be returning the real
internal index of the item id appearing

        # within the final_zappinds csv file.

        self.RatingsNumber = ratings_number

        # Columns number is the number of columns within the complete ratings file.

        self.ColumnsNumber = columns_number

        #Before setting any of the internal parameters load unique rated items.

        self.load_rated_items_ids()

        # Create a dictionary map connecting the values stored within the RatedItems list with
the corresponding indices matrix I.

        self.ItemsNumber = len(self.RatedItems)

        I = range(0,self.ItemsNumber,1)

        self.ItemIndex = dict(zip(self.RatedItems,I))

        self.UsersNumber = users_number

        self.CatchupRecordsNumber = catchup_records_number


        self.VodRecordsNumber = vod_records_number

        print "Internal parameters were successfully set"

    def
set_ratings_memmap_files_parameters(self,users_file,items_file,categories_file,ratings_file,t
imestamps_file):

        # This method sets all memmap related files names that are required in order to store
the vast volume of

        # rating data.

        self.UsersFile = users_file

        self.ItemsFile = items_file

        self.CategoriesFile = categories_file

        self.TimestampsFile = timestamps_file

        print "Memmap files related parameters were successfully set"

    def
set_users_baseline_time_instance_terms_memmap_files_parameters(self,baseline_terms_d
```

```
ata_directory):

      self.BaselineTermsDataDirectory = baseline_terms_data_directory

      self.UsersBaselineTimeInstanceTermsFiles = []

      for current_user_index in range(0,self.UsersNumber):

      filename = baseline_terms_data_directory + 'user_' + str(current_user_index+1) +
'_baseline_time_instance_terms.dat'

      self.UsersBaselineTimeInstanceTermsFiles.append(filename)

      #print self.UsersBaselineTimeInstanceTermFiles

    def
set_users_vectors_time_instance_terms_memmap_files_parameters(self,users_vectors_ter
ms_data_directory):

      self.UsersVectorsTermsDataDirectory = users_vectors_terms_data_directory

      self.UsersVectorsTimeInstanceTermsFiles = []

      for current_user_index in range(0,self.UsersNumber):

      filename = users_vectors_terms_data_directory + 'user_' + str(current_user_index+1)
+ '_vectors_time_instance_terms.dat'

      self.UsersVectorsTimeInstanceTermsFiles.append(filename)

      #print self.UsersVectorsTimeInstanceTermsFiles

    def set_users_indices_memmap_files_parameters(self,users_data_directory):

      # This method sets the relative paths for all user-indices related memmap files.

      # Calling this method is essential since it provide access to the user-indices related
data

      # by retrieving the contents of the corresponding files.

      self.UsersDataDirectory = users_data_directory



      current_user_index_file = self.UsersDataDirectory + 'user_' + str(current_user_index)
+ '_indices.dat'

      self.UsersIndicesFiles.append(current_user_index_file)

      #print self.UsersIndicesFiles

    def
set_cross_validation_memmap_files_parameters(self,cross_validation_data_directory):
```

```
        self.CrossValidationDataDirectory = cross_validation_data_directory

        self.UsersTrainingIndicesFiles = []

        self.UsersTestingIndicesFiles = []

        for current_user_index in range(0,self.UsersNumber):

        #empty_matrix = []

        empty_matrix = np.array([],dtype='|S9')

        self.UsersTrainingIndicesFiles.append(empty_matrix)

        for current_fold_index in range(0,self.FoldsNumber):

          training_filename = cross_validation_data_directory + 'user_' +
str(current_user_index+1) + '_fold_' + str(current_fold_index+1) + '_train_indices.dat'

          testing_filename = cross_validation_data_directory + 'user_' +
str(current_user_index+1) + '_fold_' + str(current_fold_index+1) + '_test_indices.dat'

          self.UsersTrainingIndicesFiles[current_user_index] =
np.hstack((self.UsersTrainingIndicesFiles[current_user_index],training_filename))

          self.UsersTestingIndicesFiles[current_user_index] =
np.hstack((self.UsersTestingIndicesFiles[current_user_index],testing_filename))

        #print self.UsersTrainingIndicesFiles

        #print self.UsersTestingIndicesFiles

      def set_users_indices_memmap_variables(self):

        # This method sets the contents of the users-indices related memmap objects.

        # The purpose of this method is to be called only once in order to save the related user
indices

        # information within the corresponding .dat file. Therefore, it should appear
commented within the

        # future versions of the main() method code.

        self.UsersFilePointer = np.memmap(self.UsersFile, dtype='int', mode='r',
shape=(1,self.RatingsNumber))

        self.UsersIds = self.UsersFilePointer[:]

        for current_user_index in range(1,self.UsersNumber+1):

        #print 'current_user_index: %d' %current_user_index

        current_user_indices = np.where(self.UsersIds==current_user_index)
```

```
        #pprint.pprint(current_user_indices)

        current_user_indices = current_user_indices[1]

        #pprint.pprint(current_user_indices)

        current_user_indices_num = np.size(current_user_indices)

        #print 'current_user_indices_num: %d' %current_user_indices_num

        print 'current_user_indices_file: %s' %self.UsersIndicesFiles[current_user_index-1]

        user_indices_file_pointer = np.memmap(self.UsersIndicesFiles[current_user_index-
1],dtype='int',mode='w+',shape=(1,current_user_indices_num))

        user_indices_file_pointer[0,:] = current_user_indices

        del self.UsersFilePointer

        self.UsersIds = None



    def
set_algorithm_parameters(self,users_ratings_percentage,features_dimensionality,items_rati
ngs_percentage,folds_number):

        # This method sets algorithm related parameters.

        # UsersRatingsPercentage: is the percentage of user's ratings that must fall within
each time interval.

        # TimeIntervalsNumber: is the corresponding number of time intervals that should be
considered.

        #       Keep in mind that TimeIntervalsNumber = round(1 / UsersRatingsPercentage).

        #       In case the pre-rounded number is not an integer the percentage residuals from

        #       the (TimeIntervalsNumber-1) intervals will be added to the last interval during
the formation

        #       of the UsersControlPoints variable. That is the last interval will take all
remaining values.

        # UsersControlPointsNumber: is the number of control points within the time-line of
interaction for each user including

        #       the starting and ending point of each time interval. Keep in mind that:

        #       usersControlPoitnsNumber = TimeIntervalsNumber + 1  and that the number
of control

        #       points is equal for all users.

        # FeaturesDimensionality: is the dimensionality of each user user- or item- related
```

```
vector.

        # ItemsRatingsPercentage: is the percentage of items' rating (in fact all ratings) that
fall within each bin.

        # BinsNumber: is the number of bins to be considered along the time-line of all
interactions stored within the

        #     complete ratings file. Keep in mind that BinsNumber = round(1 /
ItemsRatingsPercentage).

        #     In case the pre-rounded number is not an integer the percentage residuals from
the (BinsNumber-1) bins

        #     will be added to the last interval during the formation of the ItemsControlPoints
variable. That is the

        #     last interval will take all remaining values.

        # ItemsControlPointsNumber: is the number of control points within the complete
time-line of interaction for all rating

        #         records available. Keep in mind that:

        #         ItemsControlPointsNumber = BinsNumber + 1

        # FoldsNumber: is the data-set splitting parameter to be used by the cross validation
scheme in order to partition the

        #     complete data-set into training and testing subsets.

        # TrainingPercentage: is the percentage of user's rating instances to be used as
training patterns in each time interval.

        #     It is very important to ensure that each time interval is equally represented
within the training

        #     and testing data so that the corresponding model parameters can be sufficiently
determined.

        #     It is clear that the number of training and testing patterns for each user will not
be the same.

        #     Keep in mind that: TrainingPercentage = 1 - (1 / FoldsNumber)

        # DaysNumber: is the number of days within the week.

        self.UsersRatingsPercentage = users_ratings_percentage

        self.TimeIntervalsNumber = int(round(1 / self.UsersRatingsPercentage))

        self.UsersControlPointsNumber = self.TimeIntervalsNumber + 1

        self.FeaturesDimensionality = features_dimensionality
```

```
        self.ItemsRatingsPercentage = items_ratings_percentage
        self.BinsNumber = int(round(1 / self.ItemsRatingsPercentage))
        self.ItemsControlPointsNumber = self.BinsNumber + 1
        self.FoldsNumber = folds_number
        self.TrainingPercentage = 1 - (1/self.FoldsNumber)
        self.TestingPercentage = 1 - self.TrainingPercentage
        self.DaysNumber = 7
        print "Algorithm related parameters were successfully set"
    def set_catchup_data(self):
        self.CatchupData = np.empty(shape=(self.CatchupRecordsNumber,3),dtype='int')
        fp = open(self.CatchupCsv,'rb')
        reader = csv.reader(fp, delimiter=',',quotechar='"',escapechar='\\')
        row_index = 0
        for row in reader:

            self.CatchupData[row_index,0] = catchup_user_id
            self.CatchupData[row_index,1] = catchup_item_id
            self.CatchupData[row_index,2] = catchup_category_id
            row_index += 1
        fp.close()
        catchup_items_ids = self.CatchupData[:,1]
        catchup_items_unique_ids = np.unique(catchup_items_ids)
        catchup_items_unique_ids_number = np.size(catchup_items_unique_ids)
        catchup_items_real_indices = range(0,catchup_items_unique_ids_number)
        self.CatchupItemIndex = dict(zip(catchup_items_ids,catchup_items_real_indices))
        print "Contents of catchup related csv data were succesfully loaded"
    def set_vod_data(self):
        self.VodData = np.empty(shape=(self.VodRecordsNumber,3),dtype='int')
```

```
        fp = open(self.VodCsv,'rb')
        reader = csv.reader(fp, delimiter=',',quotechar='"',escapechar='\\')
        row_index = 0
        for row in reader:
        vod_user_id = int(row[0])
        vod_item_id = int(row[1])
        vod_category_id = int(row[2])
        self.VodData[row_index,0] = vod_user_id
        self.VodData[row_index,1] = vod_item_id
        self.VodData[row_index,2] = vod_category_id
        row_index += 1
        fp.close()
        vod_items_ids = self.VodData[:,1]
        vod_items_unique_ids = np.unique(vod_items_ids)
        vod_items_unique_ids_number = np.size(vod_items_unique_ids)


        self.VodItemIndex = dict(zip(vod_items_ids,vod_items_real_indices))
        print "Contents of vod related csv data were succesfully loaded"
    def get_user_catchup_items_ids(self,current_user_index):
        # current_user_index argument should be within the [1..UsersNumber] interval.


        current_user_indices = np.where(users_indices==current_user_index+1)
        current_user_catchup_items_ids = self.CatchupData[current_user_indices,1]
        current_user_catchup_items_ids =
self.get_real_catchup_items_ids(current_user_catchup_items_ids)
        return current_user_catchup_items_ids
    def get_user_catchup_items_vectors(self,user_catchup_items_ids):
        catchup_items_number = np.size(user_catchup_items_ids)
        if(catchup_items_number==0):
```

```
        user_catchup_items_vectors = np.random.randn(0,0)
        else:
        user_catchup_items_vectors = self.CatchupItemsVectors[user_catchup_items_ids,:]
        return user_catchup_items_vectors

    def set_user_catchup_items_vectors(self,user_catchup_items_ids,Ycatch):

        if(catchup_items_num!=0):
        self.CatchupItemsVectors[user_catchup_items_ids,:] = Ycatch
        def set_user_vod_items_vectors(self,user_vod_items_ids,Yvod):

        if(vod_items_num):
        self.VodItemsVectors[user_vod_items_ids,:] = Yvod
    def get_user_vod_items_ids(self,current_user_index):
        # current_user_index argument should be within the [1..UsersNumber] interval.
        users_indices = self.VodData[:,0]
        current_user_indices = np.where(users_indices==current_user_index+1)
        current_user_vod_items_ids = self.VodData[current_user_indices,1]
        current_user_vod_items_ids =
self.get_real_vod_items_ids(current_user_vod_items_ids)
        return current_user_vod_items_ids
    def get_user_vod_items_vectors(self,user_vod_items_ids):
        vod_items_number = np.size(user_vod_items_ids)
        if(vod_items_number==0):
        user_vod_items_vectors = np.random.randn(0,0)
        else:
        user_vod_items_vectors = self.VodItemsVectors[user_vod_items_ids,:]
        return user_vod_items_vectors
    def initialize_memmap_variables(self):
```

```
        # This method initializes the memmap related file pointer variable for reading.

        # Keep in mind that when file memmap file pointers are set with writing mode the
contents of

        # corresponding variables are initialized with zeros.

        self.UsersFilePointer = np.memmap(self.UsersFile, dtype='int', mode='w+',
shape=(1,self.RatingsNumber))

        self.ItemsFilePointer = np.memmap(self.ItemsFile, dtype='int', mode='w+',
shape=(1,self.RatingsNumber))

        self.CategoriesFilePointer = np.memmap(self.CategoriesFile, dtype='int', mode='w+',
shape=(1,self.RatingsNumber))

        self.TimestampsFilePointer = np.memmap(self.TimestampsFile, dtype='int64',
mode='w+', shape=(1,self.RatingsNumber))

        print "File pointers to memmap files were successfully set"



    def set_memmap_variables(self):

        # This method loads the contents of ratings csv file into the corresponding memmap-
related variable files.

        # Keep in mind that for this method to work properly it is necessary to previously call
the

        # "initialize_memmap_variables" method.

        # File pointers are deleted at the end of this method. Therefore, future reading
operations require

        # additional initialization by enforcing the reading mode of operation.

        # Keep also in mind that this method is intended to run only once. Therefore, it should
appear commented

        # in future versions of the main() method code.

        fp = open(self.RatingsCsv,'rb')

        reader = csv.reader( fp, delimiter=',', quotechar='"', escapechar='\\' )

        row_index = 0

        for row in reader:

        user_id = int(row[0])

        item_id = int(row[1])
```

```
        category_id = int(row[2])
        rating = float(row[3])
        timestamp = long(row[4])
        self.UsersFilePointer[0,row_index] = user_id
        self.ItemsFilePointer[0,row_index] = item_id
        self.TimestampsFilePointer[0,row_index] = timestamp
        row_index += 1
        fp.close()
        del self.UsersFilePointer
        del self.ItemsFilePointer
        del self.CategoriesFilePointer
        del self.RatingsFilePointer
        del self.TimestampsFilePointer
        print "Contents of memmap objects-related variables were successfully set"
    def check_memmap_variables(self):
        # This method tests whether the contents of the ratings csv file have been successfully loaded.
        # The process consists in the following steps:
        # Each memmap file is loaded in a corresponding numpy array stored in memory.
        # Each array is subsequently printed.
        # Each file pointer is deleted.
        # Each numpy array is set to None so that the corresponding memory space is deallocated.
        self.UsersFilePointer = np.memmap(self.UsersFile, dtype='int', mode='r', shape=(1,self.RatingsNumber))
        self.UsersIds = self.UsersFilePointer[0,:]
        pprint.pprint(self.UsersIds)
        del self.UsersFilePointer
        self.UsersIds = None
        self.ItemsFilePointer = np.memmap(self.ItemsFile, dtype='int', mode='r',
```

```
shape=(1,self.RatingsNumber))

        self.ItemsIds = self.ItemsFilePointer[0,:]

        pprint.pprint(self.ItemsIds)

        del self.ItemsFilePointer

        self.ItemsIds = None

        self.CategoriesFilePointer = np.memmap(self.CategoriesFile, dtype='int', mode='r',
shape=(1,self.RatingsNumber))

        self.CategoriesIds = self.CategoriesFilePointer[0,:]

        pprint.pprint(self.CategoriesIds)

        del self.CategoriesFilePointer

        self.CategoriesIds = None

        self.RatingsFilePointer = np.memmap(self.RatingsFile, dtype='float32', mode='r',
shape=(1,self.RatingsNumber))

        self.Ratings = self.RatingsFilePointer[0,:]

        pprint.pprint(self.Ratings)

        self.Ratings = None

        self.TimestampsFilePointer = np.memmap(self.TimestampsFile, dtype='int64',
mode='r', shape=(1,self.RatingsNumber))

        self.Timestamps = self.TimestampsFilePointer[0,:]

        pprint.pprint(self.Timestamps)

        del self.TimestampsFilePointer

        self.Timestamps = None

        print "Contents of memmap objects-related variable were successfuly loaded"

    def check_user_indices_memmap_variables(self):

        # This method checks the contents of the user-indices related memmap files.

        # Keep in mind that before retrieving the contents of the variables the corresponding
file pointers

        # must be set with reading mode. When reading mode is on it is not mandatory to set
the shape of

        # the corresponding array.
```

```
        for current_user_index in range(1,self.UsersNumber+1):

        user_indices_file_pointer = np.memmap(self.UsersIndicesFiles[current_user_index-
1],dtype='int',mode='r')

        current_user_indices = user_indices_file_pointer[:]

        print 'user_indices[%d]:' %current_user_index



        del user_indices_file_pointer

    def
get_control_point_position(self,control_point_index,ratings_number,ratings_per_interval):

        # This function returns the position of a control point within the time-line of a
particular user

        # given the corresponding index and the number of ratings per interval.

        # The time-line of a particular user is the chronologically sorted collection of time-
stamps when

        # ratings of that particular user were given.

        # The position corresponds to the actual array position starting counting from 0.

        # Always keep in mind that the number of ratings are the same for all interval within

        # the time-line of a particular user with a potential exception regarding the last one
which

        # might be the larger in the case when UsersRatingsNumber(u) cannot be divided into
equal partitions

        # of size UsersRatingsNumber(u) * UsersRatingsPercentage.

        # Also mind that the variable control_point_index takes values within the interval
[0..ControlPointsNumber-1].

        # Moreover, it is critical to understand that there exist two kinds of control points. The
first kind

        # involves the control points coinciding with the values of the starting and ending
positions of the time-line

        # of a particular user. The second kind of control points are the internal control points
within a user's time-line

        # storing the time-stamp value of the last time-instance in a particular interval.

        if(control_point_index==0):

        control_point_position = 0
```

```
        elif(control_point_index==self.UsersControlPointsNumber-1):

        control_point_position = ratings_number - 1

        else:

        control_point_position = (control_point_index * ratings_per_interval) - 1

        return control_point_position

    def
get_time_interval_start_position(self,time_interval_index,ratings_number,ratings_per_interv
al):

        # This function returns the starting position of a given time_interval_index within the
time-line of a given user.

        # Mind that the variable time_interval_index takes values in the interval
[1..TimeIntervalNumber].

        # The starting position of a given interval, designated by the variable
time_interval_index, can be determined

        # relatively to the position of the control_point_index previous to the current
time_interval_index. This particular

        # control_point_index is called starting_control_point_index. In this setting, the
starting position of the given

        # time_interval_index is the next position relatively to the position of the
starting_control_point_index. The exception

        # concerns the first time interval (time_interval_index == ending_control_point_index
== 1) when the starting position

        # of the interval coincides with the position of the starting_control_point_index.

        starting_control_point_index = time_interval_index - 1

        ending_control_point_index = time_interval_index

        if(ending_control_point_index==1):

        time_interval_start_position =
self.get_control_point_position(starting_control_point_index,ratings_number,ratings_per_int
erval)

        else:

        time_interval_start_position =
self.get_control_point_position(starting_control_point_index,ratings_number,ratings_per_int
erval)+1
```

```
    def
get_time_interval_end_position(self,time_interval_index,ratings_number,ratings_per_interv
al):

        # This function returns the starting position of a given time_interval_index within the
time-line of a given user.

        # Mind that the variable time_interval_index takes values in the interval
[1..TimeIntervalNumber].

        # The ending position of a given time interval. designated by the variable
time_interval_index, can be determined by

        # locating the position of the control_point_index which is equal to the current
time_interval_index. This particular

        # control_point_index is called ending_control_point_index.

        ending_control_point_index = time_interval_index

        time_interval_end_position =
self.get_control_point_position(ending_control_point_index,ratings_number,ratings_per_inte
rval)

        return time_interval_end_position



        # This function initializes two place holders (lists), namely TrainElements and
TestElements, storing the user time-stamps

        # indices that are used for training and testing respectively in each fold. These lists are
to be incrementally populated

        # by processing the time line of each user time-interval by interval. That is, the cross
validation process involves splitting

        # each time interval into corresponding training and testing partitions. In this setting
TrainElements and TestElements

        # will be formed by the aggregation of the training and testing elements of all time
intervals within the time-line of a

        # particular user.

        self.TrainElements = []

        self.TestElements  = []

        for current_fold_index in range(0,self.FoldsNumber):

        #empty_matrix = []

        empty_matrix = np.array([],dtype='int')
```

```
        self.TrainElements.append(empty_matrix)
        self.TestElements.append(empty_matrix)
    def update_interval_train_elements(self,interval_values,time_interval_index):
        interval_size = np.size(interval_values)
        testing_size = round(self.TestingPercentage * interval_size)
        interval_indices = np.array(range(0,interval_size))
        #print 'interval_indices:'
        #print interval_indices
        for current_fold_index in range(0,self.FoldsNumber):
        #print 'current_fold_index: %d' %(current_fold_index+1)
        start_testing_position = current_fold_index * testing_size
        end_testing_position = ((current_fold_index+1)*testing_size-1)
        if(current_fold_index!=self.FoldsNumber-1):
          testing_indices = interval_indices[start_testing_position:end_testing_position+1]
        else:
          testing_indices = interval_indices[start_testing_position:]
        training_indices = np.setdiff1d(interval_indices,testing_indices)
        train_elements = interval_values[training_indices]
        #print "train elements size: %d" %np.size(train_elements)
        self.TrainElements[current_fold_index] =
np.hstack((self.TrainElements[current_fold_index],train_elements))
        #print 'TrainElements current length: %d'
%len(self.TrainElements[current_fold_index])
        if(time_interval_index==1 and current_fold_index==0):
          print type(training_indices)
          print "training_indices:"
          print training_indices
          print "Current Train Elements from temporary variables:"
          print train_elements
```

```
        print "Current Train Elements:"
        print self.TrainElements[current_fold_index]
      train_elements = None
    def update_interval_test_elements(self,interval_values,time_interval_index):
      interval_size = np.size(interval_values)
      testing_size = round(self.TestingPercentage * interval_size)
      interval_indices = np.array(range(0,interval_size))
      print "interval size: %d" %np.size(interval_indices)
      #print 'interval_indices:'
      #print interval_indices
      for current_fold_index in range(0,self.FoldsNumber):
      #print 'current_fold_index: %d' %(current_fold_index+1)
      start_testing_position = current_fold_index * testing_size
      end_testing_position = ((current_fold_index+1)*testing_size-1)
      if(current_fold_index!=self.FoldsNumber-1):
        testing_indices = interval_indices[start_testing_position:end_testing_position+1]
      else:
        testing_indices = interval_indices[start_testing_position:]
      #print "test elements size %d" %np.size(test_elements)
      self.TestElements[current_fold_index] =
np.hstack((self.TestElements[current_fold_index],test_elements))
      #print 'TestElements current length: %d' %len(self.TestElements[current_fold_index])
      if(time_interval_index==1 and current_fold_index==0):
        print type(testing_indices)
        print "testing_indices"
        print testing_indices
        print "Current Test Elements from temporary variables:"
        print test_elements
        print "Current Test Elements:"
```

```
        print self.TestElements[current_fold_index]
      test_elements = None
    def update_interval_train_test_elements(self,interval_values,time_interval_index):
      interval_size = np.size(interval_values)
      testing_size = round(self.TestingPercentage * interval_size)
      interval_indices = np.array(range(0,interval_size))
      print "interval size: %d" %np.size(interval_indices)
      #print 'interval_indices:'
      #print interval_indices
      for current_fold_index in range(0,self.FoldsNumber):
      #print 'current_fold_index: %d' %(current_fold_index+1)
      start_testing_position = current_fold_index * testing_size
      end_testing_position = ((current_fold_index+1)*testing_size-1)
      if(current_fold_index!=self.FoldsNumber-1):
        testing_indices = interval_indices[start_testing_position:end_testing_position+1]
      else:
        testing_indices = interval_indices[start_testing_position:]
      training_indices = np.setdiff1d(interval_indices,testing_indices)
      train_elements = interval_values[training_indices]
      test_elements = interval_values[testing_indices]
      #print "train elements size: %d" %np.size(train_elements)
      #print "test elements size %d" %np.size(test_elements)
      #self.XXX[current_fold_index].append(train_elements)
      #self.YYY[current_fold_index].append(test_elements)


      self.TestElements[current_fold_index].append(test_elements)
      #print 'TrainElements current length: %d'
%len(self.TrainElements[current_fold_index])
      #print 'TestElements current length: %d' %len(self.TestElements[current_fold_index])
```

```
        if(time_interval_index==1 and current_fold_index==0):
          print "training_indices:"
          print training_indices
          print "testing_indices"
          print testing_indices
          print "Current Train Elements from temporary variables:"


          print "Current Test Elements from temporary variables:"
          print test_elements


          print self.TrainElements[current_fold_index]
          print "Current Test Elements:"
          print self.TestElements[current_fold_index]
      def concatenate_train_test_elements(self):
        #self.report_train_test_elements_components_sizes()
        for current_fold_index in range(0,self.FoldsNumber):
        self.TrainElements[current_fold_index] =
sum(self.TrainElements[current_fold_index],[])
        self.TestElements[current_fold_index] =
sum(self.TestElements[current_fold_index],[])


      def set_cross_validation_memmap_variables(self):
        # This method sets the training and testing indices per fold and user.
        # Mind that these sets of indices are subsets of the corresponding sets of indices for
each user.
        self.TimestampsFilePointer =
np.memmap(self.TimestampsFile,dtype='int64',mode='r',shape=(1,self.RatingsNumber))
        self.Timestamps = self.TimestampsFilePointer[:]
        for current_user_index in range(1,self.UsersNumber+1):
        #print 'current_user_index: %d' %current_user_index
```

```
        user_indices_file_pointer = np.memmap(self.UsersIndicesFiles[current_user_index-
1],dtype='int',mode='r')

        current_user_indices = user_indices_file_pointer[:]

        current_user_timestamps = self.Timestamps[0,current_user_indices]

        sorted_timestamps_indices = np.argsort(current_user_timestamps)

        sorted_current_user_indices = current_user_indices[sorted_timestamps_indices]

        ratings_number = np.size(sorted_current_user_indices)

        print 'ratings_number: %d' %ratings_number

        ratings_per_interval = math.floor(ratings_number*self.UsersRatingsPercentage)

        print 'intervals_number: %d' %self.TimeIntervalsNumber

        self.initialize_train_test_elements()

        print self.TrainElements

        print self.TestElements

        #self.report_train_test_elements_components_sizes()

        for time_interval_index in range(1,self.TimeIntervalsNumber+1):

          print 'time_interval_index: %d' %time_interval_index

          time_interval_start_position =
self.get_time_interval_start_position(time_interval_index,ratings_number,ratings_per_interv
al)

          time_interval_end_position =
self.get_time_interval_end_position(time_interval_index,ratings_number,ratings_per_interv
al)

          print 'time_interval_start_position: %d' %time_interval_start_position


          time_interval_user_indices =
sorted_current_user_indices[time_interval_start_position:time_interval_end_position+1]

          self.update_interval_train_elements(time_interval_user_indices,time_interval_index)

          self.update_interval_test_elements(time_interval_user_indices,time_interval_index)


#self.update_interval_train_test_elements(time_interval_user_indices,time_interval_index)

        self.report_train_test_elements_components_sizes()
```

```
        #self.concatenate_train_test_elements()
        for current_fold_index in range(0,self.FoldsNumber):
          current_fold_train_indices = self.TrainElements[current_fold_index]
          current_fold_train_indices_num = np.size(current_fold_train_indices)
          current_fold_test_indices = self.TestElements[current_fold_index]
          current_fold_test_indices_num = np.size(current_fold_test_indices)


          print self.UsersTestingIndicesFiles[current_user_index-1][current_fold_index]
          current_user_fold_train_indices_file_pointer =
np.memmap(self.UsersTrainingIndicesFiles[current_user_index-
1][current_fold_index],dtype='int',mode='w+',shape=(1,current_fold_train_indices_num))
          current_user_fold_test_indices_file_pointer =
np.memmap(self.UsersTestingIndicesFiles[current_user_index-
1][current_fold_index],dtype='int',mode='w+',shape=(1,current_fold_test_indices_num))
          current_user_fold_train_indices_file_pointer[0,:] = current_fold_train_indices
          current_user_fold_test_indices_file_pointer[0,:] = current_fold_test_indices
          del current_user_fold_train_indices_file_pointer
          del current_user_fold_test_indices_file_pointer
        del user_indices_file_pointer
        del self.TimestampsFilePointer
        self.Timestamps = None
      def report_train_test_elements_components_sizes(self):
        print "Training elements components report:"
        for current_fold_index in range(0,self.FoldsNumber):
        print "ELEMENTS IN FOLD %d = %d"
%(current_fold_index+1,np.size(self.TrainElements[current_fold_index]))
        #element_index = 1
        #for element in self.TrainElements[current_fold_index]:
          #print "ELEMENT %d SIZE = %d" %(element_index,np.size(element))
          #element_index += 1
```

```
        print "Testing elements components report:"

        for current_fold_index in range(0,self.FoldsNumber):

        print "ELEMENTS IN FOLD %d = %d"
%(current_fold_index+1,np.size(self.TestElements[current_fold_index]))

        #element_index = 1

        #for element in self.TestElements[current_fold_index]:

          #print "ELEMENT %d SIZE = %d" %(element_index,np.size(element))

          #element_index += 1



        UsersFoldsTrainingNumber =
np.empty(shape=(self.UsersNumber,self.FoldsNumber),dtype='int')

        UsersFoldsTestingNumber =
np.empty(shape=(self.UsersNumber,self.FoldsNumber),dtype='int')

        for current_user_index in range(1,self.UsersNumber+1):

        for current_fold_index in range(0,self.FoldsNumber):

          current_user_fold_train_file = self.UsersTrainingIndicesFiles[current_user_index-
1][current_fold_index]

          current_user_fold_test_file = self.UsersTestingIndicesFiles[current_user_index-
1][current_fold_index]

          current_user_fold_train_indices_file_pointer =
np.memmap(current_user_fold_train_file,dtype='int',mode='r')

          current_user_fold_test_indices_file_pointer =
np.memmap(current_user_fold_test_file,dtype='int',mode='r')

          current_user_fold_train_indices = current_user_fold_train_indices_file_pointer[:]

          current_user_fold_test_indices = current_user_fold_test_indices_file_pointer[:]

          current_user_fold_train_indices_num = np.size(current_user_fold_train_indices)

          current_user_fold_test_indices_num = np.size(current_user_fold_test_indices)

          UsersFoldsTrainingNumber[current_user_index-1,current_fold_index] =
current_user_fold_train_indices_num

          UsersFoldsTestingNumber[current_user_index-1,current_fold_index] =
current_user_fold_test_indices_num

          print current_user_fold_train_indices
```

```
        print current_user_fold_test_indices

        del current_user_fold_train_indices_file_pointer

        del current_user_fold_test_indices_file_pointer

      for current_user_index in range(1,self.UsersNumber+1):

      for current_fold_index in range(0,self.FoldsNumber):

        print 'user: %d fold: %d train instances: %d'
%(current_user_index,current_fold_index+1,UsersFoldsTrainingNumber[current_user_index
-1,current_fold_index])

        print 'user: %d fold: %d test instances: %d'
%(current_user_index,current_fold_index+1,UsersFoldsTestingNumber[current_user_index-
1,current_fold_index])

    def allocate_algorithm_variables(self):

      # This method allocates all variables required by the program but not the learning
algorithm oer se.

      self.UsersRatingsNumber = np.empty(shape=(1,self.UsersNumber),dtype='int')

      self.UsersStartingPoints = np.empty(shape=(1,self.UsersNumber),dtype='int64')

      self.UsersEndingPoints = np.empty(shape=(1,self.UsersNumber),dtype='int64')

      self.UsersControlPoints =
np.empty(shape=(self.UsersNumber,self.UsersControlPointsNumber),dtype='int64')

      self.ItemsControlPoints =
np.empty(shape=(1,self.ItemsControlPointsNumber),dtype='int64')

      self.UsersFoldsTrainingNumber =
np.empty(shape=(self.UsersNumber,self.FoldsNumber),dtype='int')

      self.UsersFoldsTestingNumber =
np.empty(shape=(self.UsersNumber,self.FoldsNumber),dtype='int')

      print "Algorithm variables were successfully allocated"

    def set_users_starting_ending_points(self):

      # This method sets the starting and ending point time-stamp for each user.

      #self.UsersFilePointer = np.memmap(self.UsersFile, dtype='int', mode='r',
shape=(1,self.RatingsNumber))

      self.TimestampsFilePointer = np.memmap(self.TimestampsFile, dtype='int64',
mode='r', shape=(1,self.RatingsNumber))

      #self.UsersIds = self.UsersFilePointer[:]
```

```
        self.Timestamps = self.TimestampsFilePointer[:]

        for current_user_index in range(1,self.UsersNumber+1):

        #print 'current_user_index = %d' %current_user_index

        user_indices_file_pointer = np.memmap(self.UsersIndicesFiles[current_user_index-
1],dtype='int',mode='r')

        current_user_indices = user_indices_file_pointer[:]

        #current_user_indices = np.where(self.UsersIds==current_user_index)

        #current_user_indices = current_user_indices[1]

        #print "current_user_indices [size=%d]:" %np.size(current_user_indices)

        #print 'shape[current_user_indices] = %d)' %np.shape(current_user_indices)

        #print current_user_indices

        #for c_u_i in current_user_indices:

          #print c_u_i

        #position_indices = current_user_indices

        #print position_indices

        current_user_timestamps = self.Timestamps[0,current_user_indices]

        self.UsersStartingPoints[0,current_user_index-1] = current_user_timestamps.min()

        self.UsersEndingPoints[0,current_user_index-1] = current_user_timestamps.max()

        del user_indices_file_pointer

        print self.UsersStartingPoints

        #del self.UsersFilePointer

        #self.UsersIds = None

        del self.TimestampsFilePointer

        self.Timestamps = None

        print "Starting and Ending points timestamps were succesfully set."

    def set_users_control_points(self):

        # This method sets the exact control points within the time-line of each user.

        #self.UsersFilePointer = np.memmap(self.UsersFile, dtype='int', mode='r',
shape=(1,self.RatingsNumber))
```

```
        self.TimestampsFilePointer = np.memmap(self.TimestampsFile, dtype='int64',
mode='r', shape=(1,self.RatingsNumber))

        #self.UsersIds = self.UsersFilePointer[:]

        self.Timestamps = self.TimestampsFilePointer[:]

        for current_user_index in range(1,self.UsersNumber+1):

        #print 'current_user_index = %d' %current_user_index

        user_indices_file_pointer = np.memmap(self.UsersIndicesFiles[current_user_index-
1],dtype='int',mode='r')

        current_user_indices = user_indices_file_pointer[:]

        current_user_indices_num = np.size(current_user_indices)

        self.UsersRatingsNumber[0,current_user_index-1] = current_user_indices_num

        #current_user_indices = np.where(self.UsersIds==current_user_index)

        #current_user_indices = current_user_indices[1]

        current_user_timestamps = self.Timestamps[0,current_user_indices]

        #print 'size[current_user_timestamps] = %d' %np.size(current_user_timestamps)

        #print 'shape[current_user_timestamps] = (%d,%d)'
%(np.shape(current_user_timestamps)[0],np.shape(current_user_timestamps)[1])

        #print 'size[sorted_timestamps] = %d' %np.size(sorted_timestamps)

        #print 'shape[sorted_timestamps] = (%d,%d)'
%(np.shape(sorted_timestamps)[0],np.shape(sorted_timestamps)[1])

        ratings_number = np.size(sorted_timestamps)

        ratings_per_interval = math.floor(ratings_number * self.UsersRatingsPercentage)

        internal_control_points = 0

        for control_point_index in range(0,self.UsersControlPointsNumber):

          if(control_point_index==0):

            self.UsersControlPoints[current_user_index-1,control_point_index] =
self.UsersStartingPoints[0,current_user_index-1]

          elif(control_point_index==self.UsersControlPointsNumber-1):

            self.UsersControlPoints[current_user_index-1,control_point_index] =
self.UsersEndingPoints[0,current_user_index-1]

          else:
```

```
            internal_control_points += 1
            control_point_position = (internal_control_points * ratings_per_interval) - 1
            #print 'control_point_position = %d' %control_point_position
            self.UsersControlPoints[current_user_index-1,control_point_index] =
sorted_timestamps[control_point_position]
        del user_indices_file_pointer
        #del self.UsersFilePointer
        #self.UsersIds = None
        del self.TimestampsFilePointer
        self.Timestamps = None
        print "Users control points were successfully set"
    def set_items_control_points(self):
        self.TimestampsFilePointer = np.memmap(self.TimestampsFile, dtype='int64',
mode='r', shape=(1,self.RatingsNumber))
        self.Timestamps = self.TimestampsFilePointer[:]
        self.ItemsStartingPoint = self.Timestamps.min()
        self.ItemsEndingPoint = self.Timestamps.max()
        self.ItemsDuration = self.ItemsEndingPoint - self.ItemsStartingPoint
        print 'ItemsStartingPoint = %d' %self.ItemsStartingPoint
        print 'ItemsEndingPoint = %d' %self.ItemsEndingPoint
        sorted_timestamps = np.sort(self.Timestamps)
        ratings_per_bin = math.floor(self.RatingsNumber * self.ItemsRatingsPercentage)
        internal_control_points = 0
        for control_point_index in range(0,self.ItemsControlPointsNumber):
        #print 'control_point_index = %d' %control_point_index
        if(control_point_index==0):
          self.ItemsControlPoints[0,control_point_index] = self.ItemsStartingPoint
        elif(control_point_index==self.ItemsControlPointsNumber-1):
          self.ItemsControlPoints[0,control_point_index] = self.ItemsEndingPoint
```

```
        else:

          internal_control_points += 1

          control_point_position = (internal_control_points * ratings_per_bin) - 1

          self.ItemsControlPoints[0,control_point_index] =
sorted_timestamps[0,control_point_position]

        del self.TimestampsFilePointer

        self.Timestamps = None

        print self.ItemsControlPoints

        print "Items control points were successfully set"

      def get_user_time_interval_index(self,user_index,timestamp):

        # This method returns the time interval index corresponding to the user and time
instance that are

        # indicated by the method's input arguments.

        # Mind that for the purposes of this method user_index is a zero-based index as well
as the returning time interval index.

        if(self.UsersControlPoints[user_index,0] <= timestamp and timestamp <=
self.UsersControlPoints[user_index,1]):

        time_interval_index = 0

        else:

        for control_point_index in range(2,self.UsersControlPointsNumber):

          if(self.UsersControlPoints[user_index,control_point_index-1] < timestamp and
timestamp <= self.UsersControlPoints[user_index,control_point_index]):

            time_interval_index = control_point_index - 1

            break



      def get_bin_index(self,timestamp):

        # This method returns the bin index corresponding to the time instance indicated by
the method's input arguments.

        # Mind that for the purposes of this method the returning bin index is zero-based.

        if(self.ItemsControlPoints[0,0] <= timestamp and timestamp <=
```

```
self.ItemsControlPoints[0,1]):

        bin_index = 0

        else:

        for control_point_index in range(2,self.ItemsControlPointsNumber):

          if(self.ItemsControlPoints[0,control_point_index-1] < timestamp and timestamp <=
self.ItemsControlPoints[0,control_point_index]):

            bin_index = control_point_index - 1

            break

        return bin_index

        def get_user_previous_vector(self,user_index,timestamp):

        # This method constructs the prev_u(t) time indicator vector which is a binary vector
of TimeIntervalsNumber

        # length that fires only on the positions of the time intervals that precede the current
time interval.

        #print "Input timestamp:%d" %timestamp

        #print "Current user starting point: %d" %self.UsersStartingPoints[0,user_index]

        #print "Current user ending point: %d" %self.UsersEndingPoints[0,user_index]

        current_time_interval = self.get_user_time_interval_index(user_index,timestamp) + 1

        previous_vector = np.zeros(shape=(1,self.TimeIntervalsNumber),dtype='int')

        # IMPORTANT: For the previous_vector indexing [0,0:current_time_interval-1] the
second argument

        # 0:current_time_interval-1 specifies number of elements, therefore number of time
intervals, where

        # we need to point until the previous time interval of the current one.

        previous_vector[0,0:current_time_interval-1] = 1

        return previous_vector


      def get_user_current_vector(self,user_index,timestamp):

        # This method constructs the curr_u(t) time indicator vector which is a binary vector of
TimeIntervalsNumber

        # length that fires only on the position of current time interval that contains the given
```

```
timestamp.

      current_time_interval_index =
self.get_user_time_interval_index(user_index,timestamp)

      current_vector = np.zeros(shape=(1,self.TimeIntervalsNumber),dtype='int')

      # IMPORTANT: For the current_vector indexing [0,current_time_interval_index] the
second argument

      # 0:current_time_interval_index specifies the index of the current time interval which
it is required

      # to be set to 1.

      current_vector[0,current_time_interval_index] = 1

      return current_vector

    def set_users_time_intervals_duration(self):

      # This method constructs the variable UsersTimeIntervalsDuration that stores for each
user and time interval the

      # corresponding time duration in days, that is 86400 seconds.

      self.UsersTimeIntervalsDuration =
np.empty(shape=(self.UsersNumber,self.TimeIntervalsNumber),dtype='float64')

      for user_index in range(0,self.UsersNumber):

      for control_point_index in range(1,self.UsersControlPointsNumber):

        current_control_point = self.UsersControlPoints[user_index,control_point_index]

        previous_control_point = self.UsersControlPoints[user_index,control_point_index-1]

        time_interval_index = control_point_index - 1

        self.UsersTimeIntervalsDuration[user_index,time_interval_index] =
time_interval_duration

      print self.UsersTimeIntervalsDuration

    def get_current_timestamp_within_interval_duration(self,user_index,timestamp):

      # This method returns the time duration in days that have elapsed between the
starting time instance of

      # the current time interval and the given timestamp.

      # This is the dev_u(t) parameter of the main learning algorithm.

      # Mind that both user_index and time_interval_index are zero-based.

      # Variable control_point_index stores the control point index which is closest from the
```

```
left from the

        # current timestamp.

        time_interval_index = self.get_user_time_interval_index(user_index, timestamp)

        control_point_index = time_interval_index

        control_point = self.UsersControlPoints[user_index,control_point_index]

        within_interval_duration = (timestamp-control_point) / 86400


    def set_users_folds_training_instances_numbers(self):

        for current_user_index in range(1,self.UsersNumber+1):

        for current_fold_index in range(0,self.FoldsNumber):

          current_user_fold_train_file = self.UsersTrainingIndicesFiles[current_user_index-
1][current_fold_index]

          current_user_fold_test_file = self.UsersTestingIndicesFiles[current_user_index-
1][current_fold_index]

          current_user_fold_train_indices_file_pointer =
np.memmap(current_user_fold_train_file,dtype='int',mode='r')

          current_user_fold_test_indices_file_pointer =
np.memmap(current_user_fold_test_file,dtype='int',mode='r')

          current_user_fold_train_indices = current_user_fold_train_indices_file_pointer[:]

          current_user_fold_test_indices = current_user_fold_test_indices_file_pointer[:]

          current_user_fold_train_indices_num = np.size(current_user_fold_train_indices)

          current_user_fold_test_indices_num = np.size(current_user_fold_test_indices)

          self.UsersFoldsTrainingNumber[current_user_index-1,current_fold_index] =
current_user_fold_train_indices_num

          self.UsersFoldsTestingNumber[current_user_index-1,current_fold_index] =
current_user_fold_test_indices_num

          del current_user_fold_train_indices_file_pointer


    def set_mean_rating(self):

        self.RatingsFilePointer =
np.memmap(self.RatingsFile,dtype='float32',mode='r',shape=(1,self.RatingsNumber))
```

```
        self.Ratings = self.RatingsFilePointer[0,:]
        self.MeanRating = np.mean(self.Ratings)
        print "MeanRating: %f" %self.MeanRating
        del self.RatingsFilePointer
        self.Ratings = None

    def set_algorithm_variables(self):
        self.set_mean_rating()
        self.set_users_starting_ending_points()
        self.set_users_control_points()
        self.set_items_control_points()

        self.set_users_folds_training_instances_numbers()
        print "Algorithm variables were successfully set"
    def allocate_learning_algorithm_variables(self):
        self.ItemsVectors = np.random.randn(self.ItemsNumber,self.FeaturesDimensionality)
        self.CatchupItemsVectors =
np.random.randn(self.CatchupNumber,self.FeaturesDimensionality)
        self.VodItemsVectors = np.random.randn(self.VodNumber,self.FeaturesDimensionality)
        self.ItemsBaselineInitialTerm = np.random.randn(1,self.ItemsNumber)
        self.ItemsBaselineBinTerms = np.random.randn(self.ItemsNumber,self.BinsNumber)

        self.UsersBaselineInitialTerm = np.random.randn(1,self.UsersNumber)
        self.UsersBaselineTimeIntervalTerms =
np.random.randn(self.UsersNumber,self.TimeIntervalsNumber)
        self.UsersVectorsInitialTerm =
np.random.randn(self.UsersNumber,self.FeaturesDimensionality)
        self.UsersVectorsTimeIntervalTerms =
np.random.randn(self.UsersNumber,self.TimeIntervalsNumber,self.FeaturesDimensionality)
        print "Learning algorithm variables were successfully allocated"
```

```
    def allocate_learning_algorithm_fold_dependent_variables(self,current_fold_index):

      self.allocate_users_baseline_time_instance_terms(current_fold_index)

      self.allocate_users_vectors_time_instance_terms(current_fold_index)

    def deallocate_learning_algorithm_fold_dependet_variables(self):

      self.deallocate_users_baseline_time_instance_terms()


    def allocate_users_vectors_time_instance_terms(self,current_fold_index):

      for current_user_index in range(0,self.UsersNumber):

      current_user_vectors_time_instance_terms_file =
self.UsersVectorsTimeInstanceTermsFiles[current_user_index]

      current_user_vectors_time_instance_terms_num =
self.UsersFoldsTrainingNumber[current_user_index,current_fold_index]

      current_user_vectors_time_instance_terms_file_pointer =
np.memmap(current_user_vectors_time_instance_terms_file,dtype='float64',mode='w+',sh
ape=(current_user_vectors_time_instance_terms_num,self.FeaturesDimensionality))

      current_user_vectors_time_instance_terms =
np.random.randn(current_user_vectors_time_instance_terms_num,self.FeaturesDimensional
ity)

      current_user_vectors_time_instance_terms_file_pointer[:,:] =
current_user_vectors_time_instance_terms

      del current_user_vectors_time_instance_terms_file_pointer

    def allocate_users_baseline_time_instance_terms(self,current_fold_index):

      for current_user_index in range(0,self.UsersNumber):

      current_user_baseline_time_instance_terms_file =
self.UsersBaselineTimeInstanceTermsFiles[current_user_index]

      current_user_baseline_time_instance_terms_num =
self.UsersFoldsTrainingNumber[current_user_index,current_fold_index]

      current_user_baseline_time_instance_terms_file_pointer =
np.memmap(current_user_baseline_time_instance_terms_file,dtype='float64',mode='w+',sh
ape=(1,current_user_baseline_time_instance_terms_num))

      current_user_baseline_time_instance_terms =
np.random.randn(1,current_user_baseline_time_instance_terms_num)


      del current_user_baseline_time_instance_terms_file_pointer
```

```
    def check_learning_algorithm_variables_allocation_dealloacation(self):

      self.allocate_learning_algorithm_variables()

      for current_fold_index in range(0,self.FoldsNumber):

      self.allocate_learning_algorithm_fold_dependent_variables(current_fold_index)

      self.deallocate_learning_algorithm_fold_dependet_variables()

      print "Testing of allocation and dealloaction of learning algorithm variables was
successful"



    def deallocate_users_baseline_time_instance_terms(self):

      for current_user_index in range(0,self.UsersNumber):

      current_user_baseline_time_instance_terms_file =
self.UsersBaselineTimeInstanceTermsFiles[current_user_index]

      if(os.path.exists(current_user_baseline_time_instance_terms_file)):

        os.remove(current_user_baseline_time_instance_terms_file)

    def deallocate_users_vectors_time_instance_terms(self):

      for current_user_index in range(0,self.UsersNumber):

      current_user_vectors_time_instance_terms_file =
self.UsersVectorsTimeInstanceTermsFiles[current_user_index]

      if(os.path.exists(current_user_vectors_time_instance_terms_file)):

        os.remove(current_user_vectors_time_instance_terms_file)

    def load_user_fold_training_indices(self,current_user_index,current_fold_index):

      # This function returns the subset of user indices that correspond to the training
instances

      # of the current user for the current fold. Current user training indices are retrieved
from

      # the corresponding .dat file where they are stored as memmap objects.

      # Mind that both current_user_index and current_fold_index are zero based.

      current_user_fold_train_indices_file_pointer =
np.memmap(self.UsersTrainingIndicesFiles[current_user_index][current_fold_index],dtype='
int',mode='r')

      current_user_fold_train_indices = current_user_fold_train_indices_file_pointer[:]
```

```
        del current_user_fold_train_indices_file_pointer

        return current_user_fold_train_indices

    def load_user_fold_timestamps(self,current_user_fold_indices):

        # This function loads from the corresponding timestamps .dat file the time instances
that are associated

        # with the current training instances that are passed as input arguments to the
method.

        self.TimestampsFilePointer = np.memmap(self.TimestampsFile,dtype='int64',mode='r')

        self.Timestamps = self.TimestampsFilePointer[:]

        current_user_fold_timestamps = self.Timestamps[current_user_fold_indices]

        self.Timestamps = None

        return current_user_fold_timestamps

    def load_user_fold_ratings(self,current_user_fold_indices):

        # This function loads from the corresponding ratings .dat file the ratings instances that
are associated

        # with the current training instances that are passed as input arguments to the
method.

        self.RatingsFilePointer = np.memmap(self.RatingsFile,dtype='float32',mode='r')

        self.Ratings = self.RatingsFilePointer[:]

        current_user_fold_ratings = self.Ratings[current_user_fold_indices]

        del self.RatingsFilePointer

        self.Ratings = None

        return current_user_fold_ratings

    def load_user_fold_items_ids(self,current_user_indices):

        # This function loads from the corresponding items ids .dat file the ratings instances
that are associated

        # with the current training instances that are passed as input arguments to the
method.

        self.ItemsFilePointer = np.memmap(self.ItemsFile,dtype='int',mode='r')



        current_user_fold_items_ids = self.ItemsIds[current_user_indices]
```

```
        del self.ItemsFilePointer

        self.ItemsIds = None

        return current_user_fold_items_ids

    def get_real_items_ids(self,items_ids):

        # This function returns the real items indices that correspond to the ones given as input arguments.

        # The returned indices are the real continuous item indices that correspond to the internal variables

        # of the algorithm.

        # IMPORTANT: This method must discriminate between the possible shapes of the input numpy array, so that

        #    traversing of items_ids should be done by items_ids[p] if shape==1 or items_ids[0,p] if shape==2

        items_ids_shape = np.shape(items_ids)

        shape_length = len(items_ids_shape)

        if(shape_length==1):

        real_items_ids = [self.ItemIndex[items_ids[p]] for p in range(0,np.size(items_ids))]

        else:

        real_items_ids = [self.ItemIndex[items_ids[0,p]] for p in range(0,np.size(items_ids))]

        return np.array(real_items_ids)

    def get_real_catchup_items_ids(self,catchup_items_ids):

        catchup_items_ids_shape = np.shape(catchup_items_ids)


        if(shape_length==1):

        real_catchup_items_ids = [self.CatchupItemIndex[catchup_items_ids[p]] for p in range(0,np.size(catchup_items_ids))]

        else:

        real_catchup_items_ids = [self.CatchupItemIndex[catchup_items_ids[0,p]] for p in range(0,np.size(catchup_items_ids))]

        return np.array(real_catchup_items_ids)

    def get_real_vod_items_ids(self,vod_items_ids):
```

```
        vod_items_ids_shape = np.shape(vod_items_ids)

        shape_length = len(vod_items_ids_shape)

        if(shape_length==1):

        real_vod_items_ids = [self.VodItemIndex[vod_items_ids[p]] for p in
range(0,np.size(vod_items_ids))]

        else:

        real_vod_items_ids = [self.VodItemIndex[vod_items_ids[0,p]] for p in
range(0,np.size(vod_items_ids))]

        return np.array(real_vod_items_ids)

    def load_user_baseline_time_instance_terms(self,current_user_index):

        # This function retrieves the user baseline time instance specific terms for the current
user passed as input

        # argument to the method. Mind that this function return the user's baseline terms for
all the training time

        # instances of the given user. Therefore, the particular time instance related user
baseline terms must

        # be subsequently retrieved.

        current_user_baseline_time_instance_terms_file =
self.UsersBaselineTimeInstanceTermsFiles[current_user_index]

        current_user_baseline_time_instance_terms_file_pointer =
np.memmap(current_user_baseline_time_instance_terms_file,dtype='float64',mode='r+')

        current_user_baseline_time_instance_terms =
current_user_baseline_time_instance_terms_file_pointer[:]

        del current_user_baseline_time_instance_terms_file_pointer

        return current_user_baseline_time_instance_terms

    def
load_user_vectors_time_instance_terms(self,current_user_index,current_fold_index):

        # This function retrieves the user vectors time instance specific terms for the current
user passed as input

        # argument to the method. Mind that this function return the user's vectors terms for
all the training time

        # instances of the given user. Therefore, the particular time instance related user
baseline terms must

        # be subsequently retrieved.
```

```
        current_user_vectors_time_instance_terms_file =
self.UsersVectorsTimeInstanceTermsFiles[current_user_index]

        current_user_vectors_time_instance_terms_num =
self.UsersFoldsTrainingNumber[current_user_index,current_fold_index]

        current_user_vectors_time_instance_terms_file_pointer =
np.memmap(current_user_vectors_time_instance_terms_file,dtype='float64',mode='r+',sha
pe=(current_user_vectors_time_instance_terms_num,self.FeaturesDimensionality))

        current_user_vectors_time_instance_terms =
current_user_vectors_time_instance_terms_file_pointer[:]

        del current_user_vectors_time_instance_terms_file_pointer

        return current_user_vectors_time_instance_terms



    def
store_user_vectors_time_instance_terms(self,current_user_index,current_fold_index,current
_user_vectors_time_instance_terms):

        # This function stores the user vectors time instance specific terms for the current user
and fold passed as

        # input arguments to the method in the corresponding .dat file. Mind that this function
stores the user's

        # vectors terms for all training time instances of the given user. Therefore, the
individual time instance

        # specific terms for the current user and fold must be first appropriately aggregated in
a temporary variable.

        current_user_vectors_time_instance_terms_file =
self.UsersVectorsTimeInstanceTermsFiles[current_user_index]

        current_user_vectors_time_instance_terms_num =
self.UsersFoldsTrainingNumber[current_user_index,current_fold_index]

        current_user_vectors_time_instance_terms_file_pointer =
np.memmap(current_user_vectors_time_instance_terms_file,dtype='float64',mode='w+',sh
ape =(current_user_vectors_time_instance_terms_num,self.FeaturesDimensionality))

        current_user_vectors_time_instance_terms_file_pointer[:] =
current_user_vectors_time_instance_terms

        del current_user_vectors_time_instance_terms_file_pointer



    def
store_user_baseline_time_instance_terms(self,current_user_index,current_fold_index,curre
```

```
nt_user_baseline_time_instance_terms):

      # This function stores the user baseline time instance specific terms for the current
user and fold passed as

      # input arguments to the method in the corresponding .dat file. Mind that this function
stores the user's

      # baseline terms for all training time instances of the given user. Therefore, the
individual time instance

      # specific terms for the current user and fold must be first appropriately aggregated in
a temporary variable.

      current_user_baseline_time_instance_terms_file =
self.UsersBaselineTimeInstanceTermsFiles[current_user_index]

      current_user_baseline_time_instance_terms_file_pointer =
np.memmap(current_user_baseline_time_instance_terms_file,dtype='float64',mode='w+',sh
ape=(1,self.UsersFoldsTrainingNumber[current_user_index,current_fold_index]))

      current_user_baseline_time_instance_terms_file_pointer[:] =
current_user_baseline_time_instance_terms

      del current_user_baseline_time_instance_terms_file_pointer

    def
print_user_vectors_time_instance_tersm(self,current_user_vectors_time_instance_terms,ti
me_instances_num):

      for time_index in range(0,time_instances_num):

      for feature_index in range(0,self.FeaturesDimensionality):

        print current_user_vectors_time_instance_terms[time_index,feature_index]

    def
compute_user_baseline_estimator(self,user_baseline_initial_term,user_baseline_time_interv
al_terms,user_baseline_time_instance_term,user_time_intervals_durations,user_previous_v
ector,user_current_vector,within_interval_duration):

      # This function computes the value of the user baseline estimator for the current user,
item and timestamp.

      # User, item and time instance related information is incorporated within the input
arguments.

      Bo = user_baseline_initial_term

      Bt = user_baseline_time_instance_term

      DT = user_time_intervals_durations

      PREV = user_previous_vector
```

```
        CURR = user_current_vector

        DEV = within_interval_duration

        #user_baseline_estimator = Bo + np.dot(PREV*DT,B) + DEV * np.dot(CURR,B) + Bt

        #user_baseline_estimator = Bo + np.dot(PREV*DT,B) + np.dot(DEV*CURR,B) + Bt

        SB = np.shape(B)

        if (LSB==1):

        user_baseline_estimator = Bo + np.dot(PREV*DT+DEV*CURR,B) + Bt

        else:

        user_baseline_estimator = Bo + np.dot(PREV*DT+DEV*CURR,B[0]) + Bt

        return user_baseline_estimator

    def
compute_item_baseline_estimator(self,item_baseline_initial_term,item_baseline_bin_term,it
em_baseline_day_term):

        # This function computes the value of the item baseline estimator for the current user,
item and timestamp.

        # User, item and time instance related information is incorporated within the input
arguments.

        Bi = item_baseline_initial_term

        Bi_bin = item_baseline_bin_term

        Bi_day = item_baseline_day_term

        item_baseline_estimator = Bi + Bi_bin + Bi_day

        return item_baseline_estimator

    def
compute_user_vector_slow(self,user_vectors_initial_term,user_vectors_time_interval_terms
,user_vectors_time_instance_term,user_previous_vector,user_current_vector,user_time_int
ervals_durations,within_interval_duration):

        # This function computes user vector for the current user, item and timestamp.

        # User, item and time instance related information is incorporated within the input
arguments.

        Po = user_vectors_initial_term # 1 - Dimensional


        Pt = user_vectors_time_instance_term # 1 - Dimensional
```

```
        PREV = user_previous_vector
        CURR = user_current_vector
        DT = user_time_intervals_durations
        user_vector = Po + Pt
        for interval_index in range(0,self.TimeIntervalsNumber):
        if(PREV[0,interval_index]==1):
          user_vector += P[interval_index,:] * DT[interval_index]
        if(CURR[0,interval_index]==1):
          user_vector += P[interval_index,:] * DEV
        if(PREV[0,interval_index]==0 and CURR[0,interval_index]==0):
          break
        return user_vector
      def
compute_user_vector(self,user_vectors_initial_term,user_vectors_time_interval_terms,user
_vectors_time_instance_term,user_previous_vector,user_current_vector,user_time_interval
s_durations,within_interval_duration):
        # This function computes user vector for the current user, item and timestamp.
        # User, item and time instance related information is incorporated within the input
arguments.
        Po = user_vectors_initial_term # 1 - Dimensional
        Pint = user_vectors_time_interval_terms # 2 - Dimensional
        Pt = user_vectors_time_instance_term # 1 - Dimensional
        PREV = user_previous_vector
        CURR = user_current_vector
        DT = user_time_intervals_durations
        user_vector = Po + Pt
        DT_PREV = DT * PREV
        DEV_CURR = DEV * CURR
        DT_DEV = DT_PREV + DEV_CURR
        R = Pint * DT_DEV
```

```
        user_vector += np.sum(R,axis=0)

        return user_vector


    def compute_user_item_estimator(self,user_vector,item_vector):

        # This function computes the user-item estimator based on the corresponding input
arguments.

        user_item_estimator = np.dot(user_vector,item_vector)

        return user_item_estimator

    def compute_item_catchup_items_estimator(self,item_vector,catchup_items_vectors):

        # This function computes the item - catchup items estimator based on the
corresponding input arguments.

        catchup_items_vectors_num = np.size(catchup_items_vectors,axis=0)

        if(catchup_items_vectors_num==0):

        item_catchup_items_estimator = 0

        else:

        Ncatch = 1 / math.sqrt(catchup_items_vectors_num)

        item_catchup_items_estimator =
np.dot(item_vector,np.sum(catchup_items_vectors,axis=0))

        item_catchup_items_estimator *= Ncatch

        return item_catchup_items_estimator

    def compute_item_vod_items_estimator(self,item_vector,vod_items_vectors):

        vod_items_vectors_num = np.size(vod_items_vectors,axis=0)

        if(vod_items_vectors_num==0):

        item_vod_items_estimator = 0

        else:

        Nvod = 1 / math.sqrt(vod_items_vectors_num)

        item_vod_items_estimator = np.dot(item_vector,np.sum(vod_items_vectors,axis=0))

        item_vod_items_estimator *= Nvod

        return item_vod_items_estimator
```

```
    def
compute_rating_estimator(self,user_baseline_estimator,item_baseline_estimator,user_item_
estimator,item_catchup_items_estimator,item_vod_items_estimator):

      rating_estimator = user_baseline_estimator + item_baseline_estimator +
user_item_estimator + item_catchup_items_estimator + item_vod_items_estimator +
self.quantize_rating_value(self.MeanRating)

      return rating_estimator

    def compute_current_training_error(self,rating,estimated_rating):

      # This function computes the training error associated with a particular training
instance.

      # These values will be aggregated in order to compute the overall training error.

      current_training_error = rating - estimated_rating

      return current_training_error

    def initialize_training_error(self):

      # This function initializes the computation of the overall training error in terms of the

      # Mean Absulute Error.

      self.TrainingError = 0

    def
update_user_baseline_initial_term(self,user_baseline_initial_term,current_training_error):

      Bo = user_baseline_initial_term

      Gamma = self.Gamma

      e = current_training_error

      Bo = Bo + Gamma * (e - Lambda * Bo)

      return Bo

    def
update_user_baseline_time_interval_terms(self,user_baseline_time_interval_terms,user_tim
e_intervals_durations,user_previous_vector,user_current_vector,within_interval_duration,cu
rrent_training_error):

      Bint = user_baseline_time_interval_terms

      DT = user_time_intervals_durations

      PREV = user_previous_vector
```

```
        CURR = user_current_vector

        DEV = within_interval_duration

        e = current_training_error

        Lambda = self.Lambda

        Bint = Bint + Gamma * (e * (DT * PREV + DEV * CURR) - Lambda * ((PREV +
CURR)*Bint))

        return Bint

    def
update_user_baseline_time_instance_term(self,user_baseline_time_instance_term,current_t
raining_error):

        Bt = user_baseline_time_instance_term

        Gamma = self.Gamma

        Lambda = self.Lambda

        e = current_training_error

        Bt = Bt + Gamma * (e - Lambda * Bt)

        return Bt

    def
update_item_baseline_initial_term(self,item_baseline_initial_term,current_training_error):

        Bi = item_baseline_initial_term

        Gamma = self.Gamma

        Lambda = self.Lambda

        e = current_training_error

        Bi = Bi + Gamma * (e - Lambda * Bi)

        return Bi

    def
update_item_baseline_bin_term(self,item_baseline_bin_term,current_training_error):

        Bi_bin = item_baseline_bin_term

        Gamma = self.Gamma

        Lambda = self.Lambda

        e = current_training_error
```

```
        Bi_bin = Bi_bin + Gamma * (e - Lambda * Bi_bin)

        return Bi_bin

    def
update_item_baseline_day_term(self,item_baseline_day_term,current_training_error):

        Bi_day = item_baseline_day_term

        Gamma = self.Gamma

        Lambda = self.Lambda

        e = current_training_error

        Bi_day = Bi_day + Gamma * (e - Lambda * Bi_day)

        return Bi_day

    def
update_user_vectors_initial_term(self,user_vectors_initial_term,item_vector,current_trainin
g_error):

        Po = user_vectors_initial_term

        Q = item_vector

        Gamma = self.Gamma

        Lambda = self.Lambda

        Po = Po + Gamma * (e * Q - Lambda * Po)

        return Po

    def
update_user_vectors_time_interval_terms(self,user_vectors_time_interval_terms,user_time
_intervals_durations,user_previous_vector,user_current_vector,within_interval_duration,ite
m_vector,current_training_error):

        Pint = user_vectors_time_interval_terms

        q = item_vector

        DT = user_time_intervals_durations

        PREV = user_previous_vector

        CURR = user_current_vector

        DEV = within_interval_duration

        e = current_training_error

        Gamma = self.Gamma
```

```
        Lambda = self.Lambda

        DT_PREV = DT * PREV

        DEV_CURR = DEV * CURR

        DT_DEV = DT_PREV + DEV_CURR

        I =
np.ones(shape=(self.TimeIntervalsNumber,self.FeaturesDimensionality),dtype='float64')

        Q = q * I

        Pint = Pint + Gamma * (e * (DT_DEV*Q) -Lambda * Pint)

        return Pint

    def
update_user_vectors_time_instance_term(self,user_vectors_time_instance_term,item_vecto
r,current_training_error):

        Pt = user_vectors_time_instance_term

        Q = item_vector

        Gamma = self.Gamma

        Lambda = self.Lambda

        e = current_training_error

        Pt = Pt + Gamma * (e * Q - Lambda * Pt)

        return Pt

    def
update_catchup_items_vectors(self,catchup_items_vectors,item_vector,current_training_err
or):

        catchup_items_vectors_num = np.size(catchup_items_vectors,axis=0)

        if(catchup_items_vectors_num==0):

        Ycatch = catchup_items_vectors

        else:

        q = item_vector


        Ncatch = 1 / math.sqrt(catchup_items_vectors_num)

        e = current_training_error
```

```
        Lambda = self.Lambda
        Gamma = self.Gamma
        I =
np.ones(shape=(catchup_items_vectors_num,self.FeaturesDimensionality),dtype='float64')
        Q = q * I
        Ycatch = Ycatch + Gamma * (Ncatch * e * Q  - Lambda * Ycatch)
        return Ycatch
    def
update_vod_items_vectors(self,vod_items_vectors,item_vector,current_training_error):
        vod_items_vectors_num = np.size(vod_items_vectors,axis=0)
        if(vod_items_vectors_num==0):
        Yvod = vod_items_vectors
        else:
        q = item_vector
        Yvod = vod_items_vectors
        Nvod = 1 / math.sqrt(vod_items_vectors_num)
        e = current_training_error
        Lambda = self.Lambda

        I =
np.ones(shape=(vod_items_vectors_num,self.FeaturesDimensionality),dtype='float64')
        Q = q * I
        Yvod = Yvod + Gamma * (Nvod * e * Q - Lambda * Yvod)
        return Yvod
    def
update_item_vector(self,item_vector,user_vector,catchup_items_vectors,vod_items_vectors
,current_training_error):
        e = current_training_error
        Gamma = self.Gamma
        Lambda = self.Lambda
```

```
        Q = item_vector
        P = user_vector
        Ycatch = catchup_items_vectors
        Yvod = vod_items_vectors
        # Compute catchup factor: (keeping in mind that catchup_items_vectors may be
empty([]))
        catchup_items_vectors_num = np.size(Ycatch,axis=0)
        if(catchup_items_vectors_num==0):
        catchup_factor = 0
        else:
        Ncatch = 1 / math.sqrt(catchup_items_vectors_num)
        catchup_factor = Ncatch * np.sum(Ycatch,axis=0)
        # Compute vod factor: (keeping in mind that vod_items_vectors may be empty([]))
        vod_items_vectors_num = np.size(Yvod,axis=0)
        if(vod_items_vectors_num==0):
        vod_factor = 0
        else:
        Nvod = 1 / math.sqrt(vod_items_vectors_num)

        # Update item vector:
        Q = Q + Gamma * (e * (P + catchup_factor + vod_factor) - Lambda * Q)
        return Q
    def update_training_error(self,current_ratings_number,current_training_error):
        # This function updates the estimation of the overall training error in terms of MAE
        # by taking into consideration the current training error and the current number of
        # ratings that have been processed.
        self.TrainingError = (((current_ratings_number-1)*self.TrainingError) +
abs(current_training_error))/current_ratings_number
    def update_convergence_status(self):
```

```
        # This function computes whether the execution of the learning algorithm should be
stopped.

        print "MAE TRAINING: %f" %self.TrainingError

        if(self.TrainingError > self.TargetTrainingError):

        return False

        else:

        return True

    def set_learning_algorithm_parameters(self,Gamma,Lambda,TargetTrainingError):

        # This function sets the learning algorithm parameters, that is Gamma and Lambda.

        self.Gamma = Gamma


        self.TargetTrainingError = 0.01


    def train_learing_algorithm(self):

        # This routine performs the actual training of the parameters induced by our model.

        # ----------------------------------------------------------------------------------------
--------

        # IMPORTANT NOTE: FOR THE MOMENT IT IS ONLY A SKETS OF THE FINAL CODE FOR
TESTING PURPOSES.

        # ADDITION VARIABLES SHOULD BE SET WITHIN THE SELF OBJECT CONTROL THE
EXECUTION OF THE LEARNING PROCESS.

        # CONSIDER A FUNCTION SUCH AS: "set_learning_algorithm_parameters(...)"

        # ----------------------------------------------------------------------------------------
--------

        print "EXECUTING LEARNING ALGORITHM TRAINING PROCESS:"

        # PREALLOCATE INTERNAL NON_FOLD DEPENDENT LEARNING ALGORITHM
PARAMETERS:

        self.allocate_learning_algorithm_variables()

        # TRAVERSE THRUGH THE VARIOUS FOLDS: (Mind that the fold index is zero-based!)

        for current_fold_index in range(0,self.FoldsNumber):

        print "TRAVERSING FOLD:%d" %(current_fold_index+1)
```

```
        # PREALLOCATE INTERNAL FOLD DEPENDENT LEARNING ALGORITHM PARAMETERS:

        self.allocate_learning_algorithm_fold_dependent_variables(current_fold_index)

        # INITIALIZE CONVERGENCE STATUS TO FALSE AND INITIALIZE OVERALL TRAINING
ERROR:

        converged = False

        self.initialize_training_error()

        # REPEAT UNTIL CONVERGENCE:

        while(not converged):

          # INITIALIZE COUNTER KEEPING TRACK OF THE OVERALL NUMBER OR RATINGS
PROCESSED.

          overall_ratings_processed = 1

          # TRAVERSE THROUGH THE VARIOUS USERS: (Mind that the user index is zero-
based!)

          for current_user_index in range(0,self.UsersNumber):

            #print "TRAVERSING USER:%d" %(current_user_index+1)

            # Load user-fold specific training indices, timestamps and ratings:

            current_user_fold_train_indices =
self.load_user_fold_training_indices(current_user_index, current_fold_index)

            current_user_fold_timestamps =
self.load_user_fold_timestamps(current_user_fold_train_indices)

            current_user_fold_ratings =
self.load_user_fold_ratings(current_user_fold_train_indices)

            current_user_fold_items_ids =
self.load_user_fold_items_ids(current_user_fold_train_indices)

            current_user_fold_items_ids =
self.get_real_items_ids(current_user_fold_items_ids)

            # Load user-specific baseline and vectors time instance terms.

            current_user_baseline_time_instance_terms =
self.load_user_baseline_time_instance_terms(current_user_index)

            current_user_vectors_time_instance_terms =
self.load_user_vectors_time_instance_terms(current_user_index,current_fold_index)

            # Load user-specific time intervals durations.

            user_time_intervals_durations =
```

```
self.UsersTimeIntervalsDuration[current_user_index,:]

        # GET USER CATCHUP AND VOD INDICES:

        # SET THESE INDICES TO [] WHEN DEACTIVATING THE INFLUENCE OF CATCHUP
AND VOD ITEMS

        #user_fold_catchup_indices = np.random.randn(0,0)

        #user_fold_vod_indices = np.random.randn(0,0)

        user_fold_catchup_indices = self.get_user_catchup_items_ids(current_user_index)

        user_fold_vod_indices = self.get_user_vod_items_ids(current_user_index)

        # GET USER CATCHUP AND VOD ITEMS VECTORS:

        user_fold_catchup_items_vectors =
self.get_user_catchup_items_vectors(user_fold_catchup_indices)

        user_fold_vod_items_vectors =
self.get_user_vod_items_vectors(user_fold_vod_indices)

        # TRAVERSE THROUGH THE VARIOUS USER-SPECIFIC / TIME INSTANCE-SPECIFIC
RATED ITEMS AND CORRESPONDING

        # TIME INSTANCES USED FOR TRAINING: (also initialize a required internal index)

        internal_index = 0

        for
(current_user_fold_train_index,current_user_fold_timestamp,current_user_fold_rating,curre
nt_user_fold_item_id) in
zip(current_user_fold_train_indices,current_user_fold_timestamps,current_user_fold_ratings
,current_user_fold_items_ids):

        # NEXT 2 LINEs TO BE REMOVED IN THE NEAR FUTURE

        if(current_user_fold_rating>1):

          current_user_fold_rating = 1

        # THIS LINE OF CODE QUANTIZES RATINGS WITHIN {1,2,3,4,5}. OMMIT IF NO
QUANTIZATION SHOULD HAPPEN.

        current_user_fold_rating = self.quantize_rating_value(current_user_fold_rating)

        #print "TRAVERSING INTERNAL TIME INSTANCE:%d" %(internal_index+1)

        # GET USER BASELINE ESTIMATOR REQUIRED VARIABLES:

        user_baseline_initial_term = self.UsersBaselineInitialTerm[0,current_user_index]

        user_baseline_time_interval_terms =
self.UsersBaselineTimeIntervalTerms[current_user_index,:]
```

```
        user_baseline_time_instance_term =
current_user_baseline_time_instance_terms[internal_index]

        user_previous_vector =
self.get_user_previous_vector(current_user_index,current_user_fold_timestamp)

        user_current_vector =
self.get_user_current_vector(current_user_index,current_user_fold_timestamp)

        within_interval_duration =
self.get_current_timestamp_within_interval_duration(current_user_index,current_user_fold_
timestamp)

        # GET USER VECTORS REQUIRED VARIABLES:

        user_vectors_initial_term = self.UsersVectorsInitialTerm[current_user_index,:]

        user_vectors_time_interval_terms =
self.UsersVectorsTimeIntervalTerms[current_user_index,:,:]

        user_vectors_time_instance_term =
current_user_vectors_time_instance_terms[internal_index,:]

        # GET ITEM VECTOR:

        item_vector = self.ItemsVectors[current_user_fold_item_id,:]

        # GET ITEM BASELINE ESTIMATOR REQURED VARIABLES:

        item_baseline_initial_term =
self.ItemsBaselineInitialTerm[0,current_user_fold_item_id]

        day_index =
datetime.datetime.fromtimestamp(current_user_fold_timestamp).weekday()

        item_baseline_bin_term =
self.ItemsBaselineBinTerms[current_user_fold_item_id,bin_index]

        item_baseline_day_term =
self.ItemsBaselineDayTerms[current_user_fold_item_id,day_index]

        # COMPUTE USER BASELINE ESTIMATOR:

        user_baseline_estimator =
self.compute_user_baseline_estimator(user_baseline_initial_term,
user_baseline_time_interval_terms, user_baseline_time_instance_term,
user_time_intervals_durations, user_previous_vector, user_current_vector,
within_interval_duration)

        #print "user baseline estimator = %f" %user_baseline_estimator

        item_baseline_estimator =
self.compute_item_baseline_estimator(item_baseline_initial_term, item_baseline_bin_term,
item_baseline_day_term)
```

```
        #print "item baseline estimator = %f" %item_baseline_estimator

        # COMPUTE USER VECTOR:

        user_vector = self.compute_user_vector(user_vectors_initial_term,
user_vectors_time_interval_terms, user_vectors_time_instance_term,
user_previous_vector, user_current_vector, user_time_intervals_durations,
within_interval_duration)

        #print user_vector

        # COMPUTE USER_ITEM ESTIMATOR:

        user_item_estimator = self.compute_user_item_estimator(user_vector,item_vector)

        #print "user_item_estimator: %f" %user_item_estimator

        # COMPUTE ITEM - CATCHUP ITEMS ESTIMATOR:

        item_catchup_items_estimator =
self.compute_item_catchup_items_estimator(item_vector,user_fold_catchup_items_vectors)

        #print "item_catchup_items_estimator: %f" %item_catchup_items_estimator

        # COMPUTE ITEM - VOD ITEMS ESTIMATOR:

        item_vod_items_estimator =
self.compute_item_vod_items_estimator(item_vector,user_fold_vod_items_vectors)

        #print "item vod items estimator: %f" %item_vod_items_estimator

        rating_estimator = self.compute_rating_estimator(user_baseline_estimator,
item_baseline_estimator, user_item_estimator, item_catchup_items_estimator,
item_vod_items_estimator)

        #print "rating estimator: %d" %rating_estimator

        #COMPUTE CURRENT (FOLD-USER-ITEM/TIME INSTANCE-SPECIFIC) TRAINING
ERROR:

        current_training_error =
self.compute_current_training_error(current_user_fold_rating,rating_estimator)

        #print "current_training_error: %f" %current_training_error

        # GRADIENT DESCENT - BASED UPDATE OF USER BASELINE INITIAL TERM:

        Bo = self.update_user_baseline_initial_term(user_baseline_initial_term,
current_training_error)

        #print "Bo: %f" %Bo

        # GRADIENT DESCENT - BASED UPDATE OF USER BASELINE TIME INTERVAL
TERMS:

        Bint  =
```

```
self.update_user_baseline_time_interval_terms(user_baseline_time_interval_terms,
user_time_intervals_durations, user_previous_vector, user_current_vector,
within_interval_duration, current_training_error)

        #print "Bint: "

        # GRADIENT DECENT - BASED UPDATE OF USER BASELINE TIME INSTANCE TERM:

        Bt =
self.update_user_baseline_time_instance_term(user_baseline_time_instance_term,
current_training_error)

        #print "Bt: "

        #print Bt

        # GRADIENT DECENT - BASED UPDATE OF ITEM BASELINE INITIAL TERM:

        Bi = self.update_item_baseline_initial_term(item_baseline_initial_term,
current_training_error)

        #print "Bi: "

        #print Bi

        # GRADIENT DECENT - BASED UPDATE OF ITEM BASELINE BIN TERM:

        Bi_bin = self.update_item_baseline_bin_term(item_baseline_bin_term,
current_training_error)

        #print "Bi_bin:"

        #print Bi_bin

        # GRADIENT DECENT - BASED UPDATE OF ITEM BASELINE DAY TERM:

        Bi_day = self.update_item_baseline_day_term(item_baseline_day_term,
current_training_error)

        #print "Bi_day:"

        #print Bi_day

        # GRADIENT DESCENT - BASED UPDATE OF USER VECTOR INITIAL TERM:

        Po = self.update_user_vectors_initial_term(user_vectors_initial_term, item_vector,
current_training_error)

        #print "Po:"

        #print Po

        # GRADIENT DESCENT - BASED UPDATE OF USER VECTOR TIME INTERVAL TERMS:

        Pint =
self.update_user_vectors_time_interval_terms(user_vectors_time_interval_terms,
```

```
user_time_intervals_durations, user_previous_vector, user_current_vector,
within_interval_duration, item_vector, current_training_error)

        #print "Pint"

        #print Pint

        # GRADIENT DESCENT - BASED UPDATE OF USER VECTOR TIME INSTANCE TERM:

        Pt =
self.update_user_vectors_time_instance_term(user_vectors_time_instance_term,
item_vector, current_training_error)

        #print "Pt:"

        #print Pt

        # COMPUTE THE NEW VERSION OF THE USER VECTOR BASED ON THE UPDATE
VERSIONS OF THE CORRESPONDING PARAMETERS:

        P = self.compute_user_vector(Po, Pint, Pt, user_previous_vector,
user_current_vector, user_time_intervals_durations, within_interval_duration)

        #print "P:"

        #print P

        # GRADIENT DESCENT - BASED UPDATE CATCHUP ITEMS VECTORS:

        Ycatch = self.update_catchup_items_vectors(user_fold_catchup_items_vectors,
item_vector, current_training_error)

        #print "Ycatch:"

        #print Ycatch

        # GRADIENT DESCENT - BASED UPDATE VOD ITEMS VECTORS:

        Yvod = self.update_vod_items_vectors(user_fold_vod_items_vectors,item_vector,
current_training_error)

        #print "Yvod:"

        #print Yvod

        # GRADIENT DESCENT - BASED UPDATE ITEMS VECTOR:

        Q = self.update_item_vector(item_vector, user_vector,
user_fold_catchup_items_vectors, user_fold_vod_items_vectors, current_training_error)

        #print "Q:"

        #print Q

        # COMPUTE UPDATED ESTIMATORS: (BASED ON THE UPDATED VERSIONS OF THE
```

```
INTERNAL PARAMETERS)

        updated_user_baseline_estimator =
self.compute_user_baseline_estimator(Bo,Bint,Bt,user_time_intervals_durations,user_previo
us_vector,user_current_vector,within_interval_duration)

        updated_item_baseline_estimator =
self.compute_item_baseline_estimator(Bi,Bi_bin,Bi_day)

        updated_user_item_estimator = self.compute_user_item_estimator(P,Q)

        updated_item_catchup_items_estimator =
self.compute_item_catchup_items_estimator(Q,Ycatch)

        updated_rating_estimator =
self.compute_rating_estimator(updated_user_baseline_estimator,updated_item_baseline_es
timator,updated_user_item_estimator,updated_item_catchup_items_estimator,updated_ite
m_vod_items_estimator)

        # COMPUTE UPDATED CURRENT TRAINING ERROR:

        updated_current_training_error =
self.compute_current_training_error(current_user_fold_rating,updated_rating_estimator)

        #print "current_training error = %f updated_current_training_error  = %f"
%(current_training_error,updated_current_training_error)

        #if(abs(current_training_error)>abs(updated_current_training_error)):

          #print "OK"

        #else:

          #print "NOT OK"

        # COMPUTE UPDATED OVERALL TRAINING ERROR:


self.update_training_error(overall_ratings_processed,updated_current_training_error)

        #print "TRAINING ERROR: %f" %self.TrainingError

        # STORE BACK UPDATED VERSIONS OF INTERNAL PARAMETERS:

        self.UsersBaselineInitialTerm[0,current_user_index] = Bo #1

        self.UsersBaselineTimeIntervalTerms[current_user_index,:] = Bint #2

        self.UsersVectorsTimeIntervalTerms[current_user_index,:,:] = Pint #5

        current_user_vectors_time_instance_terms[internal_index,:] = Pt #6

        self.ItemsBaselineInitialTerm[0,current_user_fold_item_id] = Bi #7

        self.ItemsBaselineBinTerms[current_user_fold_item_id,bin_index] = Bi_bin #8
```

```
            self.ItemsBaselineDayTerms[current_user_fold_item_id,day_index] = Bi_day #9
            self.ItemsVectors[current_user_fold_item_id,:] = Q #10
            self.set_user_catchup_items_vectors(user_fold_catchup_indices, Ycatch) #11
            self.set_user_vod_items_vectors(user_fold_vod_indices, Yvod) #12
            # UPDATE INTERNAL AND EXTERNAL COUNTER VARIABLES:
            internal_index += 1
            overall_ratings_processed += 1
            # IMPORTANT!!! ONLY FOR CURRENT USE SO THAT THE CORRESPONDING .DAT
FILES CAN BE DELETED.
            # THIS IS WHERE ADDTIONAL STORSAGE OPERATIONS SHOULD BE DONE:
            current_user_baseline_time_instance_terms = None
            current_user_vectors_time_instance_terms = None
            user_vectors_time_instance_term = None
           #self.Gamma *= 2
        self.deallocate_learning_algorithm_fold_dependet_variables()
        print "LEARNING ALGORITHM EXECUTION COMPLETED"


    def main():
        # File test_new.csv is only a portion of the complete ratings matrix. It is only used for
testing purposes.
        # The complete file that should be used is the final_zappings.csv.
        ratings_csv = 'recommendation_data/final_zappings_50_best.csv'
        catchup_csv = 'recommendation_data/final_catchup.csv'


        # IMPORTANT!!!
        # Keep in mind that not all items ids appear within the final_zappings csv file.
        # Therefore, we need an additional index list that stores the indices of all items
        # that actually appear within the ratings record. This list is saved in file
items_on_Zappings.csv.
```

```
    rated_items_csv = 'recommendation_data/items_on_Zappings.csv'

    # Set external parameters

    recommender = Recommender(ratings_csv,catchup_csv,vod_csv,rated_items_csv)

    # Set internal parameters.

    # WARNING!!! INTERNAL PARAMETER VALUES ARE USER DEFINED!!! MAKE SURE THAT THE CORRECT VALUES ARE GIVEN!!!

    # The complete number of ratings is 222906207. However, when testing on a portion of the complete ratings record

    # only a number of 241437 ratings.

    # ratings_number = 222906207

    #ratings_number = 237977

    ratings_number = 1359831

    columns_number = 5

    #users_number = 57826

    users_number = 50

    catchup_number = 7845

    vod_number = 401

    vod_records_number = 105166


recommender.set_internal_parameters(ratings_number,columns_number,users_number, catchup_number, vod_number,catchup_records_number,vod_records_number)

    # Set algorithm related parameters.

    # These parameters are very important since the final results will be highly dependent on them.

    # A good set of parameters should be experimentally determined.

    users_ratings_percentage = 1.0

    features_dimensionality = 5

    items_ratings_percentage = 1.0

    folds_number = 5

    recommender.set_algorithm_parameters(users_ratings_percentage, features_dimensionality, items_ratings_percentage,folds_number)
```

```
    # Set memmap related files parameters.
    # Mind that this list is yet to be completed.
    users_file = 'recommendation_data/users.dat'
    items_file = 'recommendation_data/items.dat'


    ratings_file = 'recommendation_data/ratings.dat'
    timestamps_file = 'recommendation_data/timestamps.dat'

recommender.set_ratings_memmap_files_parameters(users_file,items_file,categories_file,ra
tings_file,timestamps_file)
    # Initialize memmap related variables. [NOT REQUIRED WHEN CORRESPONDING DAT
FILES ALREADY EXIST!!!]
    # recommender.initialize_memmap_variables()
    # Set memmap related variables. [NOT REQUIRED WHEN CORRESPONDING DAT FILES
ALREADY EXIST!!!]
    # recommender.set_memmap_variables()
    recommender.set_catchup_data()
    recommender.set_vod_data()
    # Check memmap related variables.
    recommender.check_memmap_variables()
    # Set the directory that stores the user indices related data.
    users_data_directory = 'user_data/'
    # Set the directory that stores the user training and testing indices related data.
    cross_validation_data_directory = 'cross_validation_data/'
    # Initialize the names of the .dat files that will be storing the user related indices.
    recommender.set_users_indices_memmap_files_parameters(users_data_directory)

recommender.set_cross_validation_memmap_files_parameters(cross_validation_data_direct
ory)
    # recommender.set_users_indices_memmap_variables()
    # recommender.check_user_indices_memmap_variables()
```

```
    #recommender.set_cross_validation_memmap_variables()

    # recommender.check_cross_validation_memmap_variables()

    # Set the directory that stores the user baseline time instance term variables.

    baseline_terms_data_directory = 'baseline_terms_data/'

    # Initialize the names of the .dat files that will be storing the fold-dependent user-related baseline time instance specific term variables.


recommender.set_users_baseline_time_instance_terms_memmap_files_parameters(baseline_terms_data_directory)

    # Set the directory that stores the user vectors time instance term variables.

    users_vectors_terms_data_directory = 'users_vectors_terms_data/'

    # Initialize the names of the .dat files that will be storing the fold-dependent user-related vectors time instance specific term variables.


recommender.set_users_vectors_time_instance_terms_memmap_files_parameters(users_vectors_terms_data_directory)

    # Allocate memory for internal algorithm variables

    recommender.allocate_algorithm_variables()

    # Set the contents of the internal algorithm variables.

    recommender.set_algorithm_variables()

    # recommender.check_learning_algorithm_variables_allocation_dealloacation()

    # Set internal learning algorithm paramaters and proceed with the training process.

    Gamma = 0.000001 # BEST GAMMA

    #Gamma = 0.00001

    Lambda = 1.0

    recommender.set_learning_algorithm_parameters(Gamma,Lambda,TargetTrainingError)

    recommender.train_learing_algorithm()


  if __name__ == "__main__":

 main()
```