Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Συστήματα συστάσεων: Αντιμετώπιση αραιών δεδομένων με παραγωγή εγγράφων χαρακτηριστικών
Ονοματεπώνυμο Φοιτητή	Ζαχαρίας Εφραιμίδης
Πατρώνυμο	Ηλίας
Αριθμός Μητρώου	ΜΠΣΠ/11060
Επιβλέπων	Γεώργιος Τσιχριντζής, Καθηγητής

Ημερομηνία Παράδοσης Μάρτιος 2014

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Γεώργιος Τσιχριντζής Καθηγητής	Χαράλαμπος Κωνσταντόπουλος Επίκουρος Καθηγητής	Άγγελος Πικράκης Λέκτορας

1 Περιεχόμενα

2 Περίληψη/Abstract

2.1 Ελληνικά

Με την εξάπλωση του Διαδικτύου δόθηκε η δυνατότητα σε κάθε ενδιαφερόμενο να επεκταθεί σε ευρύτερο κοινό. Άμεσα ξεκίνησε ο σχεδιασμός και η ανάπτυξη ολοκληρωμένων συστημάτων που να επιτρέπουν την απομακρυσμένη πρόσβαση των χρηστών σε κάθε είδους δεδομένα. Σύντομα παρήχθη πλεονάζουσα πληροφορία καθιστώντας αρκετά συστήματα δυσλειτουργικά, αφού οι χρήστες δυσκολεύονταν πλέον να εντοπίσουν ενδιαφέροντα στοιχεία. Έκτοτε έχει ξεκινήσει η αναζήτηση λύσεων, που να επιτρέπουν την περαιτέρω ανάπτυξη συστημάτων, χωρίς όμως να αποθαρρύνονται οι χρήστες τους. Τα συστήματα συστάσεων είναι μία από τις προτεινόμενες κατευθύνσεις προς τον στόχο αυτό, καθώς σκοπεύουν, μέσω διαφόρων τεχνικών, να προβλέψουν τον βαθμό αποδοχής κάθε στοιχείου από κάθε χρήστη και να κάνουν τις καταλληλότερες προτάσεις. Αν και οι τεχνικές ποικίλουν, ο βασικός τρόπος λειτουργίας σχετίζεται με τα υπάρχοντα δεδομένα του συστήματος, δηλαδή βασίζεται στα χαρακτηριστικά των στοιχείων ή των χρηστών, καθώς και τις μεταξύ τους αλληλεπιδράσεις, ώστε να προτείνει κατάλληλα στοιχεία στον εκάστοτε χρήστη. Ένα από τα σημαντικότερα προβλήματα των συστημάτων συστάσεων είναι τα αραιά δεδομένα. Δηλαδή παρατηρείται η έλλειψη μεγάλου ποσοστού από τα προαναφερθέντα χαρακτηριστικά, άλλοτε λόγω πρωτοεμφανιζόμενου χρήστη, όπου το σύστημα δεν έχει στη μνήμη του αρκετές πληροφορίες, και άλλοτε λόγω εσφαλμένης μοντελοποίησης των στοιχείων, με αποτέλεσμα η δομή τους να περιέχει κενές τιμές στα περιττά χαρακτηριστικά. Στην παρούσα εργασία για τη διαπίστωση αυτού του φαινομένου πραγματοποιείται μία πειραματική υλοποίηση αξιοποιώντας δεδομένα κινηματογραφικών ταινιών από MovieLens και IMDb. Στη συνέχεια επιδιώκεται η επίλυση του προβλήματος αυτού εξάγοντας θέματα από το σύνολο των χαρακτηριστικών με χρήση του αλγορίθμου latent Dirichlet allocation και περιγράφοντας τα στοιχεία ως μείγματα των θεμάτων.

2.2 English

The proliferation of the Internet has allowed every person to extend to a wider audience. The design and development of integrated systems that allowed users to remotely access any kind of data began. Soon redundant information was produced making systems quite dysfunctional, since most users had difficulty to find interesting data. Since then research has started for solutions which should allow further development of systems, but without discouraging their users. One of the suggested guidelines for this objective is the idea of recommender systems, which, through various techniques, try to predict the degree of acceptance of each item for each user and make appropriate recommendations. Although there is a broad range of techniques available, their basic function is associated with the system's existing data, i.e. they are based on the features of the item or the user, and the interactions between them, in order to propose appropriate items to each user. One of the major problems of recommender systems is the data sparsity. A large proportion of the aforementioned features is missing, either because of the newcomer user, for whom the system has no memory, or due to incorrect modeling of the items, which makes their structure to consist of empty features. In order to confirm the existence of this problem an experimental solution is being developed by utilizing movies' data from MovieLens and IMDb. Then the paper seeks to address this problem by extracting topics from the set of the features, using the latent Dirichlet allocation algorithm, and describing items as mixtures of topics.

3 Εισαγωγή

Με την εξάπλωση του διαδικτύου δόθηκε η δυνατότητα παροχής προϊόντων και υπηρεσιών σε μεγαλύτερη κλίμακα. Όμως, η ευρύτερη ποικιλία αγαθών προκαλεί προβλήματα στις επιλογές των καταναλωτών, κάνοντας την εύρεση της καταλληλότερης λύσης για κάθε άτομο δυσκολότερη. Το πρόβλημα δυσχεραίνεται περισσότερο αν αναλογιστεί κανείς ότι η προώθηση των αγαθών γίνεται σε ένα, επίσης ευρύτερο, κοινό του οποίου οι απαιτήσεις και οι ανάγκες ποικίλουν. Γι' αυτόν τον λόγο ξεκίνησαν να αναπτύσσονται συστήματα, που να βοηθούν στη λήψη αποφάσεων προτείνοντας κατάλληλα στοιχεία στους χρήστες τους. Παραδείγματα σχετικών δραστηριοτήτων αποτελούν τα εξής:

- η αγορά ενός προϊόντος έναντι άλλων,
- η εκδρομή προς έναν από τους διαθέσιμους ταξιδιωτικούς προορισμούς,
- η προτίμηση μεταξύ των εστιατορίων μίας πόλης ή ακόμα και
- η παρακολούθηση ενός από τους τίτλους μίας ταινιοθήκης.

Τα ανωτέρω συστήματα βασιζόμενα στην πρότερη γνώση, η οποία σχετίζεται με τις επιλογές των χρηστών τους (users) ή με τα χαρακτηριστικά των στοιχείων τους (items), προσπαθούν να προβλέψουν ποια στοιχεία θα επιλέξει κάθε χρήστης στο μέλλον [1]. Κατά την επιλογή ενός στοιχείου λαμβάνεται υπόψη, εκτός από την αντικειμενική του αξία, και η χρησιμότητά του (utility) ως προς τον χρήστη. Οι βαθμοί χρησιμότητας πρέπει να αποτιμώνται βάσει συγκεκριμένων κριτηρίων για κάθε χρήστη ξεχωριστά και να ταξινομούνται από τον καλύτερο προς τον χειρότερο. Η διαδικασία η οποία προβλέπει τη χρησιμότητα κάθε επιλογής και προτείνει μία κατάλληλα ταξινομημένη λίστα ενός υποσυνόλου των στοιχείων στον αντίστοιχο χρήστη ονομάζεται σύσταση (recommendation) και κάθε εργαλείο λογισμικού ή τεχνική που μπορεί να πραγματοποιεί αυτήν τη διαδικασία λέγεται σύστημα συστάσεων (recommender system) .

Τα κριτήρια, βάσει των οποίων αποτιμούνται οι βαθμοί χρησιμότητας, σχετίζονται με τα χαρακτηριστικά των ίδιων των στοιχείων ή/και των χρηστών. Για παράδειγμα ένας εργοδότης κατά την επιλογή προσωπικού μπορεί να αξιολογήσει τους υποψήφιους υπαλλήλους κρίνοντας από τα χαρακτηριστικά των βιογραφικών τους. Αντίστοιχα, είναι δυνατόν να κάνει προσλήψεις κρίνοντας τα χαρακτηριστικά των προηγούμενων εργοδοτών κάθε υπάλληλου και τις αξιολογήσεις τους εν γένει, όπως άλλωστε συμβαίνει με τις συστατικές επιστολές. Δηλαδή τα συστήματα συστάσεων μπορούν να χωριστούν σε τρεις βασικές κατηγορίες:

- βάσει των χαρακτηριστικών των στοιχείων (content-based filtering),
- συνεργατικού φιλτραρίσματος (collaborative filtering), δηλαδή βάσει των στοιχείων που επέλεξαν άλλοι συγκεκριμένοι χρήστες, αλλά και
- βάσει συνδυασμού των δύο προηγουμένων κατηγοριών (hybrid).

Εκτός από τις προαναφερθείσες βασικές κατηγορίες έχουν αναπτυχθεί και άλλες ειδικές τεχνικές κάποιες από τις οποίες αναλύονται στο επόμενο κεφάλαιο.

Από τα παραπάνω προκύπτει ότι το πρόβλημα της παροχής συστάσεων ορίζεται ως η προσέγγιση της απόκρισης ενός χρήστη στα νέα στοιχεία, βάσει των πληροφοριών που είναι αποθηκευμένα στο σύστημα, και η πρόταση των νέων στοιχείων, που έχουν υψηλή προβλεπόμενη απόκριση. Ο τρόπος με τον οποίο ένας χρήστης μπορεί να αποκριθεί ποικίλει ανάλογα με την εφαρμογή και εμπίπτει σε τρεις κατηγορίες:

- κλιμακούμενη (scalar),
- δυαδική (binary) και
- μοναδιαία (unary).

Οι κλιμακούμενες αποκρίσεις, γνωστές και ως βαθμολογίες (ratings), αποτελούνται από πολλαπλές τιμές οι οποίες δείχνουν το επίπεδο αποδοχής ενός στοιχείου από τον χρήστη, π.χ. 1-5. Οι δυαδικές αποκρίσεις αποτελούνται από μόνο δύο τιμές που δείχνουν την αποδοχή ή μη του στοιχείου. Τέλος, οι μοναδιαίες αποκρίσεις βασίζονται στην υπόθεση ότι ο χρήστης επιλέγει μόνο στοιχεία που τον ενδιαφέρουν, οπότε σε αυτές τις εφαρμογές σημειώνονται μόνο τα

επιλεγμένα στοιχεία, αγνοώντας τη διαφορά μεταξύ των μη επιλεγμένων και των μη αποδεχτών στοιχείων.

Παρακάτω εισάγονται οι απαραίτητοι συμβολισμοί για τον τυπικό ορισμό της βασικής λειτουργίας των συστημάτων προτάσεων. Το σύνολο των χρηστών συμβολίζεται με το U, το σύνολο των στοιχείων με το I και οι πιθανές τιμές κάθε βαθμολογίας συμβολίζονται από το σύνολο S. Με το r_{ui} εννοείται η βαθμολογία του χρήστη u για το στοιχείο i. Με I_u, όπου $u \in U$, εννοείται το υποσύνολο των στοιχείων που έχουν επιλεγεί από τον χρήστη u. Αντίστοιχα με U_i, όπου $i \in I$, εννοείται το υποσύνολο των χρηστών που έχουν επιλέξει το στοιχείο i. Σκοπός είναι να βρεθεί συνάρτηση χρησιμότητας $f: U \times I \to S$ τέτοια ώστε για έναν χρήστη $u \in U$ και ένα νέο στοιχείο $j \in I - I_u$, η $f(u,j)$ να προσεγγίζει τη βαθμολογία r_{ui}. Έπειτα αρκεί να αξιοποιηθεί αυτή η συνάρτηση για να προταθεί στον τρέχοντα χρήστη u_a ένα στοιχείο i^* που να δίνει τη μέγιστη τιμή:

$$i^* = \arg\max_{j \in I - I_u} f(u_a, j)$$

Τα συστήματα συστάσεων δεν μπορούν να εγγυηθούν την απόλυτη αποδοχή των προτεινόμενων στοιχείων από όλους τους χρήστες, λόγω της ποικιλομορφίας των διαφόρων συνόλων δεδομένων που καθιστά ουσιαστικά αδύνατη την εύρεση της καταλληλότερης επιλογής για κάθε χρήστη. Επιπλέον, από ψυχολογικής άποψης, οι βαθμολογίες που εισάγουν οι ίδιοι χρήστες για τα ίδια στοιχεία σε διαφορετικές χρονικές στιγμές δεν είναι απαραίτητα οι ίδιες, άρα δεν είναι και απόλυτα προβλέψιμες. Παρόλα αυτά κάθε αποδοτικό σύστημα παραμένει χρήσιμο, αφού γενικά η πληθώρα των επιλογών μπορεί να περιπλέξει τους χρήστες, με αποτέλεσμα ο βαθμός ικανοποίησης του χρήστη (user satisfaction) να ήταν σαφώς μικρότερος αν διάλεγε στοιχεία χωρίς τη βοήθεια ενός συστήματος συστάσεων.

Ένα σημαντικό πρόβλημα που παρατηρείται είναι τα αραιά δεδομένα (sparse data), τόσο στον πίνακα των βαθμολογιών όσο και στον πίνακα των χαρακτηριστικών. Ο χρήστης δεν δύναται να αξιολογήσει όλα τα στοιχεία, αντιθέτως, δοκιμάζει πολύ λίγα στοιχεία καθ' όλη τη χρήση ενός τέτοιου συστήματος με αποτέλεσμα να δημιουργείται ένας πολύ αραιός πίνακας, δηλαδή χωρίς τιμές στα περισσότερα πεδία. Αντίστοιχα, όταν δεν χρησιμοποιούνται κοινά χαρακτηριστικά για να μοντελοποιηθούν όλα τα στοιχεία, ο πίνακας των χαρακτηριστικών μπορεί επίσης να είναι αραιός δυσχεραίνοντας την εύρεση ομοιοτήτων μεταξύ των στοιχείων.

Ο σκοπός της εργασίας αφορά την περιγραφή μίας προσέγγισης κατά την οποία κάθε στοιχείο μοντελοποιείται ως ένα μείγμα από θέματα, ώστε να μειωθεί το πρόβλημα των αραιών δεδομένων. Συγκεκριμένα προτείνεται να συλλέγονται τα χαρακτηριστικά των στοιχείων υπό μορφή κειμένων και να σχηματίζεται ένα σώμα από αυτά, δηλαδή κάθε στοιχείο να αντιστοιχίζεται με ένα σύνολο από έγγραφα. Έπειτα, το σώμα μοντελοποιείται σε θέματα, τα οποία χρησιμοποιούνται ώστε να σχηματίσουν ένα μείγμα, δηλαδή μία κοινή και πυκνή δομή με την οποία να μπορούν να περιγράφονται τα στοιχεία.

Η εργασία δομείται παρακάτω ως εξής: στο κεφάλαιο 4 παρουσιάζονται οι υπάρχουσες προσεγγίσεις, στο κεφάλαιο 5 περιγράφεται η προετοιμασία της πειραματικής υλοποίησης με χρήση δεδομένων μίας ταινιοθήκης, στο κεφάλαιο 6 παρουσιάζονται τα πειραματικά αποτελέσματα, στο κεφάλαιο 7 περιγράφεται η προτεινόμενη μεθοδολογία και στο κεφάλαιο 8 γίνεται καταγραφή των συμπερασμάτων.

4 Υπάρχουσες προσεγγίσεις

Τα συστήματα συστάσεων έχουν ολοένα αυξανόμενη παρουσία σε εμπορικές και μη εφαρμογές, όπως το IMDB, το Amazon, το NetFlix, το TripAdvisor κ.α. Αυτό έχει οδηγήσει σε διαφορετικές προσεγγίσεις υλοποίησης συστημάτων, που να προτείνουν στους χρήστες τους προϊόντα και υπηρεσίες.

Γενικότερα η παραγωγή συστάσεων από ένα σύστημα βασίζεται σε ένα συνδυασμό των παρακάτω χαρακτηριστικών του:

- Το είδος των δεδομένων που υπάρχουν στη βάση δεδομένων (π.χ. βαθμολογίες, πληροφορίες χρηστών, χαρακτηριστικά και περιεχόμενα των στοιχείων, κοινωνικές συσχετίσεις μεταξύ των χρηστών και πληροφορίες θέσεις).
- Ο αλγόριθμος που χρησιμοποιείται (π.χ. δημογραφικό, βάσει περιεχομένου, συνεργατικό, βάσει κοινωνικότητας, βάσει γνώσης και υβριδικό).
- Το επιλεγμένο μοντέλο (π.χ. άμεση χρήση των δεδομένων (memory-based) ή χρήση παραγόμενου μοντέλου από τα δεδομένα (model-based)).
- Οι χρησιμοποιημένες τεχνικές, όπως πιθανολογικές προσεγγίσεις, Μπεϋζιανά δίκτυα, αλγόριθμος κοντινότερων γειτόνων, αλγόριθμοι εμπνευσμένοι από τη βιολογία π.χ. νευρωνικά δίκτυα και γενετικοί αλγόριθμοι, ασαφή μοντέλα, τεχνικές μείωσης του βαθμού αραίωσης κ.α.
- Ο βαθμός αραίωσης της βάσης δεδομένων και ο επιθυμητός βαθμός κλιμάκωσης.
- Η απόδοση του συστήματος.
- Ο αντικειμενικός σκοπός του (π.χ. προβλέψεις και οι Ν καλύτερες προτάσεις).
- Η επιθυμητή ποιότητα των αποτελεσμάτων (π.χ. προτεραιότητα σε νέα αποτελέσματα, ποικιλία ή ακρίβεια).

4.1 Αρχή λειτουργίας

Από τα παραπάνω ξεχωρίζει το χαρακτηριστικό του χρησιμοποιούμενου αλγόριθμου καθώς καθορίζει τις εσωτερικές λειτουργίες των συστημάτων συστάσεων. Όπως προαναφέρθηκε, ένας γενικός τρόπος λειτουργίας των συστημάτων συστάσεων μπορεί να μοντελοποιηθεί ως μία συνάρτηση $f(u,i)$, που προβλέπει τον βαθμό χρησιμότητας ενός στοιχείου i για έναν χρήστη u. Στην απλούστερη της μορφή, δηλαδή στην περίπτωση που δε διατίθενται άλλες πληροφορίες πέρα από τις επιλογές των χρηστών, θα μπορούσε απλά να χρησιμοποιηθεί η συχνότητα επιλογής του στοιχείου από τους υπόλοιπους χρήστες, καθώς με αυτόν τον τρόπο αυξάνεται η πιθανότητα χρησιμότητας για τον μέσο χρήστη.

Βέβαια η απόδοση κάθε συστήματος μπορεί να κυμαίνεται, αφού υπάρχει η περίπτωση ένα σύστημα να συμπεριφέρεται με άλλο τρόπο σε ένα διαφορετικό σύνολο δεδομένων. Για παράδειγμα τα συστήματα συνεργατικού φιλτραρίσματος έχει παρατηρηθεί ότι λειτουργούν καλύτερα σε σύνολα δεδομένων όπου ο αριθμός των χρηστών είναι σημαντικά μεγαλύτερος από τον αριθμό των στοιχείων. Για αυτόν τον λόγο πρέπει να γίνεται προσεχτική επιλογή της μετρικής που χρησιμοποιείται σε κάθε σύστημα, ώστε να βελτιστοποιείται η απόδοση του για το είδος του συνόλου δεδομένων στο οποίο θα λειτουργεί.

Οι μετρικές που χρησιμοποιούνται συνήθως αφορούν την ακρίβεια (accuracy) και σχετίζονται με το μέσο σφάλμα (average error). Οι μετρικές χρησιμοποιούνται τόσο για τον υπολογισμό των αποτελεσμάτων αλλά και για τον έλεγχο της αποδοτικότητας της μεθόδου. Για το δεύτερο σκοπό, οι βαθμολογίες R χωρίζονται σε ένα εκπαιδευτικό σύνολο R_{train}, πάνω στο οποίο εκπαιδεύεται το σύστημα ώστε να μάθει τη συνάρτηση χρησιμότητας f, και σε ένα δοκιμαστικό σύνολο R_{test}, στο οποίο στη συνέχεια ζητείται να εξεταστεί η ακρίβεια των προβλέψεων [2]. Δύο δημοφιλείς μετρικές είναι το Μέσο Απόλυτο Σφάλμα (Mean Absolute Error):

$$MAE(f) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} |f(u,i) - r_{ui}|$$

και η Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error):

$$RMSE(f) = \sqrt{\frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} (f(u,i) - r_{ui})^2}$$

που είναι κατάλληλες για τη μέτρηση της ακρίβειας των προβλέψεων, αν και χάνουν τα χαρακτηριστικά που σχετίζονται με την ικανοποίηση του χρήστη [3].

Στη βιβλιογραφία έχουν προταθεί διάφορες τεχνικές που συνυπολογίζουν την ποιότητα των συστάσεων, όπως την αποφυγή των σοβαρών λαθών με χρήση μετρικών «υποστήριξης αποφάσεων», ή την αντιμετώπιση των συστάσεων ως λίστες αντί για απλές προβλέψεις, ή ακόμα και την υποβολή πολλαπλών βαθμολογιών από τους χρήστες για κάθε στοιχείο. Παρόλα αυτά για τους σκοπούς της παρούσας εργασίας αξιοποιούνται οι μετρικές μέσου σφάλματος.

4.2 Τύποι συστημάτων

Αν και η βασική τεχνική αρκεί ώστε να δώσει καλύτερα αποτελέσματα από την τυχαία επιλογή στοιχείων των χρηστών, η χρησιμότητά της είναι μικρή. Προσπαθώντας για τη βελτίωση της απόδοσης των προβλέψεων έχουν αναπτυχθεί διάφοροι αλγόριθμοι, βάσει των οποίων συχνά γίνεται η εξής κατηγοριοποίηση:

α) συνεργατικού φιλτραρίσματος (collaborative filtering),

β) βάσει γνώσης (knowledge-based),

γ) βάσει περιεχομένου (content-based),

δ) δημογραφικά (demographic),

στ) βάσει κοινωνικότητας (social-based), και

ζ) υβριδικά (hybrid) [1] [4].

4.2.1 Συνεργατικό φιλτράρισμα

Οι μέθοδοι του συνεργατικού φιλτραρίσματος προωθούν προτάσεις στοιχείων στους χρήστες χωρίς την ανάγκη επιπλέον πληροφοριών για τα στοιχεία ή τους χρήστες. Η βασική τους αρχή έγκειται στην υπόθεση ότι αν δύο χρήστες αξιολογήσουν αρκετά στοιχεία με παρόμοιο τρόπο, τότε η κοινή τους συμπεριφορά θα συνεχιστεί και σε άλλα στοιχεία [5].

Οι τεχνικές συνεργατικού φιλτραρίσματος χρησιμοποιούν βάση δεδομένων με τις προτιμήσεις στοιχείων από τους χρήστες, ώστε να προβλέψουν στοιχεία που μπορεί να αρέσουν σε ένα νέο χρήστη. Σε ένα τυπικό σενάριο, υπάρχει μία λίστα με m χρήστες $\{u_1, u_2, \dots, u_m\}$ και μία λίστα από n στοιχεία $\{i_1, i_2, \dots, i_n\}$, και κάθε χρήστης, u_i, έχει μία λίστα στοιχείων, Iu_i, τα οποία έχουν αξιολογηθεί άμεσα, από τον χρήστη, ή έμμεσα, από τη συμπεριφορά του.

Για να επιτευχθεί η παραγωγή προτάσεων, τα συστήματα συνεργατικού φιλτραρίσματος πρέπει να συσχετίσουν τις δύο διαφορετικές οντότητες: τα στοιχεία και τους χρήστες. Γι' αυτόν τον σκοπό υπάρχουν δύο τεχνικές: η βασισμένη στη μνήμη (memory-based) και η βασισμένη σε μοντέλο (model-based).

Οι πρώτες εστιάζουν σε σχέσεις μεταξύ των στοιχείων ή, εναλλακτικά, μεταξύ των χρηστών, χρησιμοποιώντας τις αξιολογήσεις των χρηστών για να υπολογίσουν τις ομοιότητες ή τις βαρύτητες μεταξύ τους. Για παράδειγμα, η συσχέτιση μεταξύ στοιχείων μοντελοποιεί την προτίμηση ενός χρήστη συγκρίνοντας τις αξιολογήσεις που ο ίδιος έχει κάνει στα στοιχεία. Μειονέκτημα αυτών των μεθόδων αποτελεί η πιθανή έλλειψη αξιολογήσεων σε κοινά στοιχεία. Αυτή η έλλειψη συντελεί στην δημιουργία αραιών δεδομένων και τα καθιστά αναξιόπιστα.

Οι δεύτερες δημιουργούν μοντέλα για την πρόβλεψη των προτιμήσεων των χρηστών χρησιμοποιώντας τα δεδομένα αξιολογήσεων. Μερικά γνωστά μοντέλα αποτελούν τα Μπεϋζιανά δίκτυα (Bayesian belief nets), τα μοντέλα συσταδοποίησης (clustering) και τα λανθάνοντα σημασιολογικά μοντέλα (latent semantic models).

4.2.2 Φιλτράρισμα βάσει γνώσης

Παραδοσιακά, τα συστήματα συστάσεων διαθέτουν δύο είδη οντοτήτων, τους χρήστες και τα στοιχεία [6]. Παρόλα αυτά, σε διάφορες εφαρμογές, είναι πραγματικά χρήσιμο να συνυπολογιστεί η παράπλευρη πληροφορία στη διαδικασία των συστάσεων, ώστε να προταθούν διαφορετικά στοιχεία στον χρήστη υπό συγκεκριμένες συνθήκες. Για παράδειγμα, μπορούν να χρησιμοποιηθούν χρονικά δεδομένα για να προταθεί ένα ταξίδι κατάλληλο για την τρέχουσα εποχή του χρόνου.

Τέτοιου είδους γνώσεις συνήθως θεωρούνται τα δεδομένα που σχετίζονται είτε με τους χρήστες είτε με τα στοιχεία και μπορούν να χρησιμοποιηθούν για τον σχηματισμό προτύπων. Με αυτόν τον τρόπο είναι δυνατό να αξιοποιηθούν τεχνικές εξόρυξης δεδομένων και περιορισμού του εύρους αναζήτησης.

Επιπλέον, μπορούν να αξιοποιηθούν πληροφορίες που δηλώνουν τον σκοπό της πρότασης, π.χ. άλλη πρόταση πρέπει να γίνει αν ο χρήστης σκοπεύει να αγοράσει ένα προϊόν για τον εαυτό του και άλλη αν πρόκειται για δώρο. Τέτοιου είδους πληροφορίες είναι ιδιαίτερα αξιοποιήσιμες σε συνδυασμό με τεχνικές μηχανικής μάθησης.

Στις βασικές γνώσεις, που χρησιμοποιούνται στα βασισμένα σε γνώση συστήματα συστάσεων, ανήκει η τοποθεσία του χρήστη σε συνδυασμό με δεδομένα για το περιβάλλον του, όπως άλλοι χρήστες ή αντικείμενα, που βρίσκονται σε κοντινή απόσταση. Αντίστοιχα χρήσιμη, όπως αναφέρθηκε και προηγουμένως, είναι η γνώση της χρονικής στιγμής, αλλά και τα διάφορα περιβαλλοντικά δεδομένα, όπως η θερμοκρασία, όταν πρόκειται για συστάσεις που επηρεάζονται από τις καιρικές συνθήκες.

4.2.3 Φιλτράρισμα βάσει περιεχομένου

Καθώς ο σκοπός των συστημάτων συστάσεων είναι να οδηγήσουν τους χρήστες σε στοιχεία που ενδιαφέρουν τον καθένα προσωπικά, ήταν λογικό να αναπτυχθούν προσεγγίσεις που αναζητούν παρόμοια στοιχεία με αυτά που αρέσουν ήδη στον καθένα ξεχωριστά. Ακριβώς αυτό πραγματοποιούν τα συστήματα προτάσεων βάσει περιεχομένου [7].

Σε αντίθεση με τα συστήματα συνεργατικού φιλτραρίσματος, που αξιοποιούν τις προτιμήσεις των άλλων χρηστών, τα συστήματα συστάσεων βάσει περιεχομένου χρησιμοποιούν τα χαρακτηριστικά των στοιχείων. Αναλυτικότερα εντοπίζουν τα στοιχεία που αρέσουν ήδη στον χρήστη και τα συγκρίνουν με τα υπόλοιπα στοιχεία. Φυσικά για να επιτευχθεί αυτό πρέπει να υπάρχει ένας τρόπος μοντελοποίησης των στοιχείων και μίας συνάρτησης υπολογισμού του βαθμού ομοιότητας μεταξύ τους.

Τα συστήματα αυτού του είδους χρησιμοποιούν διάφορες μεθόδους για τη μοντελοποίηση των στοιχείων. Όταν είναι δυνατόν να εξαχθούν χαρακτηριστικά των οποίων οι τιμές είναι συγκεκριμένες, δηλαδή τα δεδομένα τους είναι δομημένα, τότε μπορούν να χρησιμοποιηθούν αλγόριθμοι μηχανικής μάθησης για τη δημιουργία του προφίλ χρήστη (user profile), που ουσιαστικά συνοψίζει τις προτιμήσεις του. Αντίθετα, συχνά συμβαίνει να χρησιμοποιούνται έγγραφα, που περιγράφουν με αδόμητο τρόπο τα περιεχόμενα των στοιχείων. Σε αυτές τις περιπτώσεις συνήθως γίνεται σημασιολογική ανάλυση των δεδομένων ώστε να ενσωματωθούν στο προφίλ.

Με λίγα λόγια μέσω του προφίλ χρήστη είναι δυνατόν τα στοιχεία να ταξινομηθούν στις κλάσεις «αρέσει» και «δεν αρέσει». Για τον σκοπό αυτό χρησιμοποιούνται συνήθως τεχνικές μηχανικής μάθησης, οι οποίες χρησιμοποιούν τα στοιχεία που έχει ήδη αξιολογήσει ο κάθε χρήστης ξεχωριστά.

Τα κυριότερα πλεονεκτήματα αυτών των συστημάτων είναι: α) η ανεξαρτησία του κάθε χρήστη, καθώς τα ευρήματα βασίζονται αποκλειστικά στις επιλογές του, β) η διαφάνεια, που παρέχουν τα συστήματα αυτά αφού είναι δυνατό να περιγραφούν ακριβώς τα κριτήρια βάσει των οποίων γίνεται η επιλογή της κάθε σύστασης, και γ) η προσαρμοστικότητα σε νέα στοιχεία, αφού η εξαγωγή των χαρακτηριστικών κάθε νέου στοιχείου γίνεται άμεσα κατά την εισαγωγή του στο σύστημα.

Στα μειονεκτήματα συγκαταλέγονται οι εξής περιορισμοί που επηρεάζουν την επίδοση των συστημάτων συστάσεων βάσει περιεχομένου: α) η εξάρτηση από την ποιότητα των

χαρακτηριστικών που έχουν εξαχθεί, β) η προσκόλληση σε συγκεκριμένο τύπο αποτελεσμάτων λόγο κακής μοντελοποίησης των στοιχείων, και γ) η αδυναμία εξυπηρέτησης νέων χρηστών για τους οποίους, όπως είναι λογικό, δεν έχει δημιουργηθεί ακόμα προφίλ χρήστη.

4.2.4 Χρήση δημογραφικών

Αυτός ο τύπος συστημάτων συστάσεων βασίζεται στο δημογραφικό προφίλ του χρήστη και στην υπόθεση ότι οι χρήστες με διαφορετικά δημογραφικά στοιχεία έχουν και διαφορετικές προτιμήσεις ή το αντίθετο. Έχουν προκύψει προσεγγίσεις που αξιοποιούν απλές αλλά αποδοτικές λύσεις που βασίζονται στα στοιχεία του χρήστη. Για παράδειγμα, λαμβάνεται υπόψη η γλώσσα ή η χώρα του για να εμφανιστούν στοιχειά που τον αφορούν περισσότερο. Αντίστοιχα οι προτάσεις μπορούν να διαφέρουν ανάλογα με την ηλικία του χρήστη.

Οι έρευνες σχετικά με την αποδοτικότητα της συγκεκριμένης μεθόδου είναι ελλιπείς, παρόλο που η χρήση τους είναι αρκετά συχνή στην αγορά.

4.2.5 Φιλτράρισμα βάσει κοινωνικότητας

Όπως και στο συνεργατικό φιλτράρισμα, που αξιοποιείται από τα μέσα της δεκαετίας του 1990, οι προτιμήσεις των άλλων χρηστών είναι το κριτήριο βάσει του οποίου προτείνονται τα σχετικά στοιχεία. Έκτοτε έχουν προκύψει πολλοί διαφορετικοί σχεδιασμοί στον τρόπο αξιοποίησης αυτής της πληροφορίας. Σήμερα, έχουν αναπτυχθεί δικτυακές κοινότητες δίνοντας την ευκαιρία για τη βελτίωση του σχεδιασμού των συστημάτων συστάσεων. Χρησιμοποιώντας τις διάφορες σχέσεις, που σχηματίζονται στα κοινωνικά δίκτυα, επιδιώκεται η αύξηση της ακρίβειας των συστάσεων [8].

4.2.6 Υβριδικές προσεγγίσεις

Τέλος, έχουν προκύψει διαφόρων ειδών υβριδικές μέθοδοι που αξιοποιούν συνδυασμούς διαφόρων προσεγγίσεων προσπαθώντας να ελαχιστοποιήσουν τα αρνητικά κάθε μεθόδου. Συχνά παρατηρείται ο συνδυασμός συνεργατικού φιλτραρίσματος με χρήση δημογραφικών ή συνεργατικού φιλτραρίσματος και φιλτράρισμα βάσει περιεχομένου. Τα υβριδικά συστήματα συνήθως εμπνέονται από τη βιολογία και τις πιθανολογικές μεθόδους αξιοποιώντας τεχνικές όπως οι γενετικοί αλγόριθμοι, τα νευρωνικά δίκτυα, τα Μπεϋζιανά δίκτυα, η συσταδοποίηση (clustering) και τα λανθάνοντα χαρακτηριστικά (latent features).

5 Πειραματική υλοποίηση

Για την πειραματική υλοποίηση των μεθόδων αυτής της εργασίας επιλέχθηκε η χρήση του παραδείγματος της ταινιοθήκης αξιοποιώντας συνδυασμό βάσεων δεδομένων για να ληφθεί πλούσια πληροφορία τόσο σχετικά με τις βαθμολογίες των χρηστών όσο και με τα χαρακτηριστικά των ταινιών. Συγκεκριμένα συνδέοντας το σύνολο δεδομένων της IMDb με το σύνολο MovieLens είναι πλέον διαθέσιμη πληροφορία σχετικά με τους συντελεστές πάνω από 3000 ταινιών μαζί με την πλοκή τους και πλήθος λέξεων-κλειδιών ενώ παράλληλα διατίθενται βαθμολογίες 6000 περίπου χρηστών των οποίων είναι γνωστά τα δημογραφικά τους δεδομένα.

Στις επόμενες παραγράφους περιγράφεται η διαδικασία εξαγωγής των χρήσιμων δεδομένων από τις δύο βάσεις και η συσχέτισή τους βάσει τίτλου ταινίας.

5.1 Εξαγωγή δεδομένων MoviesLens

Η ανάκτηση του συνόλου δεδομένων MovieLens έγινε από τον ιστότοπο του ερευνητικού προγράμματος GroupLens, το οποίο αποτελείται από μία ομάδα του τμήματος Πληροφορικής και Μηχανικής του πανεπιστημίου της Μινεσότα με επικεφαλής τους καθηγητές John Riedl και Joseph Konstan. Το GroupLens διαθέτει ελεύθερα για λήψη διάφορα σύνολα δεδομένων, τρία εκ των οποίων αφορούν βαθμολογίες ταινιών που συλλέχθηκαν στον ιστότοπο του MovieLens (http://movielens.umn.edu). Για τους σκοπούς της παρούσας εργασίας επιλέχθηκε το σύνολο MovieLens 1M [13], που αποτελείται από 1.000.209 βαθμολογίες από 6.040 χρήστες για περίπου 3.900 ταινίες.

Το παρεχόμενο αρχείο είναι σε συμπιεσμένη μορφή zip και περιέχει 4 αρχεία κειμένου, από τα οποία το 1 περιέχει γενικές πληροφορίες σχετικά με το σύνολο των δεδομένων. Τα υπόλοιπα 3 αρχεία περιέχουν πληροφορίες βαθμολογιών, χρηστών και ταινιών αντίστοιχα και οι δομές τους περιγράφονται στη συνέχεια χρησιμοποιώντας τη γραμματική Augmented BNF (ABNF) [14], η οποία αναλύεται στο ΠΑΡΑΡΤΗΜΑ Α.

5.1.1 Περιγραφή αρχείου βαθμολογιών

Όλες οι βαθμολογίες περιέχονται στο αρχείο «ratings.dat» και έχουν την ακόλουθη δομή:

```
βαθμολογίαΧρήστη = κωδικόςΧρήστη "::" κωδικόςΤαινίας "::" βαθμολογία "::" χρονοσφραγίδα
κωδικόςΧρήστη = 1*DIGIT
κωδικόςΤαινίας = 1*DIGIT
βαθμολογία = "0" / "1" / "2" / "3" / "4" / "5"
χρονοσφραγίδα = 1*DIGIT
```

Παρατηρήσεις:

- οι κωδικοί των χρηστών περιέχουν τιμές από 1 μέχρι 6040
- οι κωδικοί των ταινιών περιέχουν τιμές από 1 μέχρι 3952
- οι βαθμολογίες αποτελούνται από 5 αστέρια (ολόκληρα μόνο)
- η χρονοσφραγίδα αναπαριστάται σε δευτερόλεπτα από το 1/1/1970
- κάθε χρήστης έχει τουλάχιστον 20 βαθμολογίες

Για αυτό το αρχείο κατασκευάστηκε ο πίνακας «ml_ratings» ως εξής:

```
CREATE TABLE `ml_ratings` (
  `UserId` decimal(4,0) NOT NULL,
  `MovieId` decimal(4,0) NOT NULL,
  `Rating` decimal(1,0) DEFAULT NULL,
  `Timestamp` decimal(10,0) DEFAULT NULL,
  PRIMARY KEY (`UserId`,`MovieId`),
  KEY `fk_ratings_users_idx` (`UserId`),
```

```
  KEY `fk_ratings_movies_idx` (`MovieId`),
  CONSTRAINT `fk_ratings_movies` FOREIGN KEY (`MovieId`) REFERENCES `ml_movies`
    (`MovieId`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_ratings_users` FOREIGN KEY (`UserId`) REFERENCES `ml_users`
    (`UserId`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

5.1.2 Περιγραφή αρχείου χρηστών

Οι πληροφορίες των χρηστών περιέχονται στο αρχείο «user.dat» και έχουν την ακόλουθη δομη:

```
χρήστης = κωδικόςΧρήστη "::" φύλο "::" ηλικία "::" επάγγελμα "::" ταχυδρομικόςΚώδικας
κωδικόςΧρήστη = 1*DIGIT
φύλο = "M" / "F"
ηλικία = "1" / "18" / "25" / "35" / "45" / "50" / "56"
επάγγελμα = ["1" / "2"] DIGIT ; ακέραιος από 0 έως 20
ταχυδρομικόςΚώδικας = 1*DIGIT ["-" 1*DIGIT]
```

Παρατηρήσεις:

- όλες οι δημογραφικές πληροφορίες παρέχονται από τους χρήστες οικιοθελώς και δεν έχουν ελεγχθεί ως προς την ακρίβειά τους. Στο παρών σύνολο δεδομένων περιέχονται μόνο χρήστες που έχουν συμπληρώσει κάποια δημογραφικά δεδομένα.
- το φύλο σημειώνεται με ένα «M» για άντρα και με ένα «F» για γυναίκα
- η ηλικία επιλέγεται από τα ακόλουθα εύρη τιμών και σημειώνεται ως εξής:
  - 1: «Under 18»
  - 18: «18-24»
  - 25: «25-34»
  - 35: «35-44»
  - 45: «45-49»
  - 50: «50-55»
  - 56: «56+»
- το επάγγελμα επιλέγεται από τις παρακάτω επιλογές:
  - 0: «other» ή δεν προσδιορίζεται
  - 1: «academic/educator»
  - 2: «artist»
  - 3: «clerical/admin»
  - 4: «college/grad student»
  - 5: «customer service»
  - 6: «doctor/health care»
  - 7: «executive/managerial»
  - 8: «farmer»
  - 9: «homemaker»
  - 10: «K-12 student»
  - 11: «lawyer»
  - 12: «programmer»
  - 13: «retired»
  - 14: «sales/marketing»

- 15: «scientist»
- 16: «self-employed»
- 17: «technician/engineer»
- 18: «tradesman/craftsman»
- 19: «unemployed»
- 20: «writer»

Για αυτό το αρχείο κατασκευάστηκε ο πίνακας «ml_users» ως εξής:

```
CREATE TABLE `ml_users` (
  `UserId` decimal(4,0) NOT NULL,
  `Gender` char(1) DEFAULT NULL,
  `AgeId` decimal(2,0) DEFAULT NULL,
  `OccupationId` decimal(2,0) DEFAULT NULL,
  `ZipCode` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`UserId`),
  KEY `fk_users_ages_idx` (`AgeId`),
  KEY `fk_users_occupations_idx` (`OccupationId`),
  CONSTRAINT `fk_users_ages` FOREIGN KEY (`AgeId`) REFERENCES `ml_ages` (`AgeId`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_users_occupations` FOREIGN KEY (`OccupationId`) REFERENCES
    `ml_occupations` (`OccupationId`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

5.1.3 Περιγραφή αρχείου ταινιών

Οι πληροφορίες των ταινιών είναι στο αρχείο «movie.dat» και έχουν την ακόλουθη δομή:

```
ταινία = κωδικόςΤαινίας "::" τίτλος "::" είδος
κωδικόςΤαινίας = 1*DIGIT
τίτλος = όνομα SP "(" έτος ")"
όνομα = 1*VCHAR
έτος = 4DIGIT
είδος = 1*VCHAR
```

Παρατηρήσεις:

- οι τίτλοι είναι πανομοιότυποι με τους τίτλους που παρέχονται από την IMDB, συμπεριλαμβάνοντας το έτος έκδοσης, παρατηρούνται όμως διαφοροποιήσεις κυρίως όσων αφορά τα άρθρα.
- τα είδη μίας ταινίας μπορεί να είναι οποιοδήποτε υποσύνολο του παρακάτω συνόλου λεκτικών διαχωριζόμενα με καθέτους «|»
  - Action
  - Adventure
  - Animation
  - Children's
  - Comedy
  - Crime
  - Documentary
  - Drama
  - Fantasy

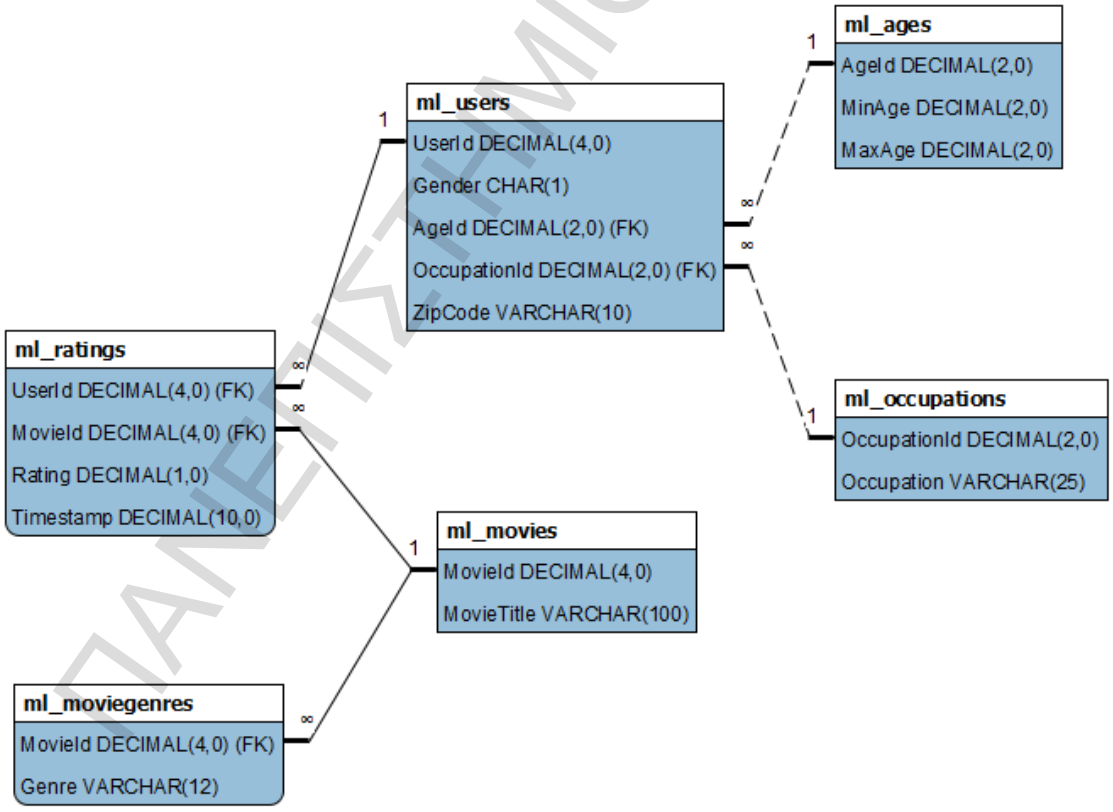
- Film-Noir
- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western

- Κάποια αναγνωριστικά ταινιών δεν αντιστοιχούν σε ταινίες λόγω εσφαλμένων διπλοεγγραφών ή/και δοκιμαστικών εισαγωγών
- Οι ταινίες εισάγονται κυρίως χειρονακτικά, επομένως μπορεί να υπάρχουν λάθη.

Για αυτό το αρχείο κατασκευάστηκε ο πίνακας «ml_movies» ως εξής:

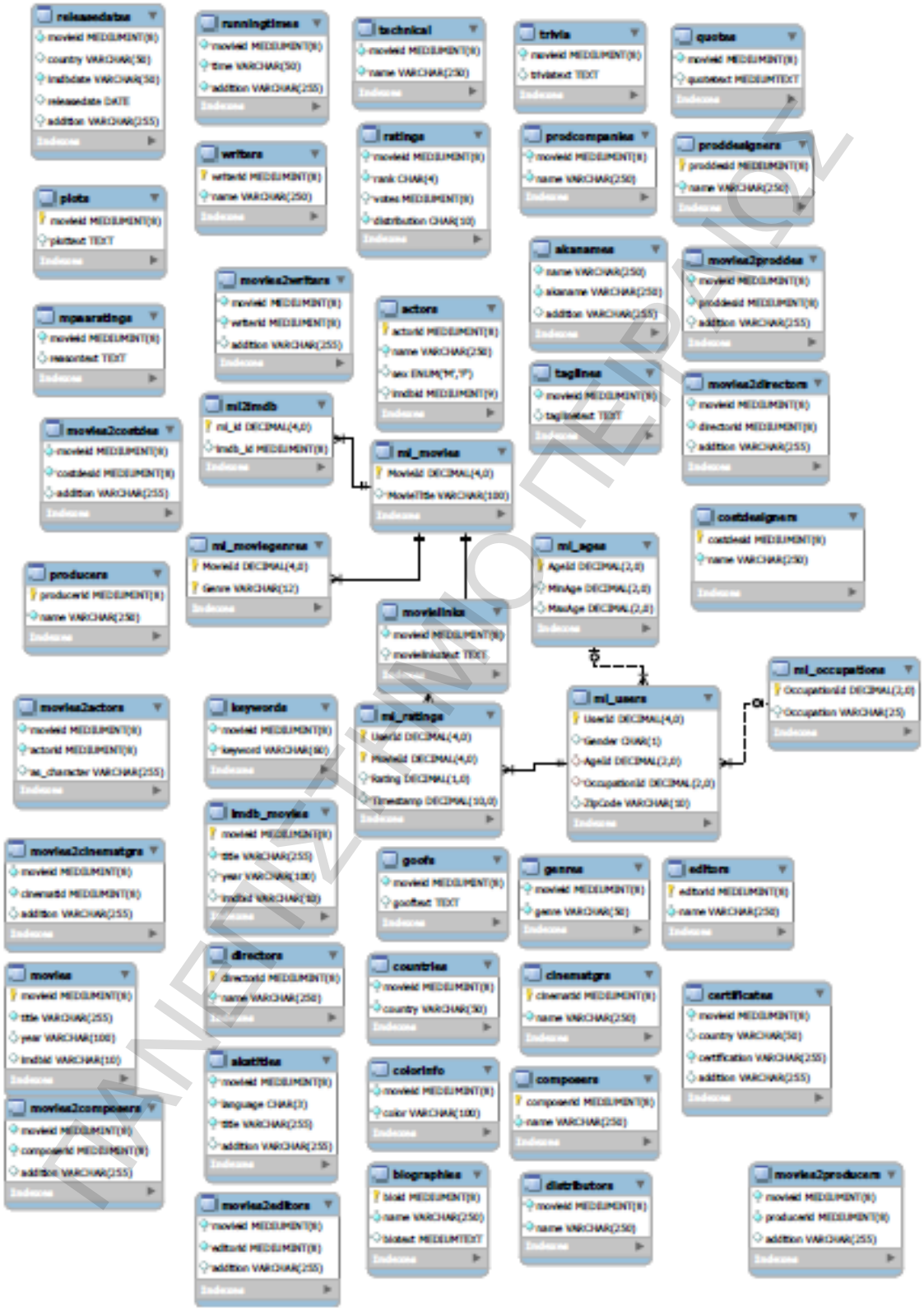
```
CREATE TABLE `ml_movies` (
  `MovieId` decimal(4,0) NOT NULL,
  `MovieTitle` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`MovieId`),
  UNIQUE KEY `MovieTitle_UNIQUE` (`MovieTitle`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Το τελικό σχήμα της βάσης μοιάζει ως εξής:



Εικόνα 1: Το σχήμα της βάσης MovieLens.

5.2 Εξαγωγή δεδομένων IMDB



Εικόνα 2: Οι πίνακες της IMDB.

Τα αρχεία-πίνακες της IMDB που χρησιμοποιούνται είναι τα εξής:

1. actors: actors.list.gz, actresses.list.gz
2. akanames: aka-names.list.gz
3. cinematgrs: cinematographers.list.gz
4. colorinfo: color-info.list.gz
5. composers: composers.list.gz
6. costdesigners: costume-designers.list.gz
7. countries: countries.list.gz
8. directors: directors.list.gz
9. editors: editors.list.gz
10. genres: genres.list.gz
11. keywords: keywords.list.gz
12. movies: movies.list.gz
13. mpaaratings: mpaa-ratings-reasons.list.gz
14. plots: plot.list.gz
15. producers: producers.list.gz
16. prodcompanies: production-companies.list.gz
17. proddesigners: production-designers.list.gz
18. quotes: quotes.list.gz
19. ratings: ratings.list.gz
20. releasedates: release-dates.list.gz
21. runningtimes: running-times.list.gz
22. taglines: taglines.list.gz
23. trivia: trivia.list.gz
24. writers: writers.list.gz

Δεν χρησιμοποιούνται σε πίνακες τα εξής αρχεία:

1. alternate-versions.list.gz
2. business.list.gz
3. complete-cast.list.gz
4. complete-crew.list.gz
5. goofs: goofs.list.gz
6. movielinks: movie-links.list.gz
7. distributors: distributors.list.gz
8. akatitles: aka-titles.list.gz
9. certificates: certificates.list.gz
10. biographies: biographies.list.gz
11. technical: technical.list.gz
12. crazy-cradits.list.gz
13. german-aka-titles.list.gz
14. iso-aka-titles.list.gz
15. italian-aka-titles.list.gz
16. language.list.gz
17. laserdisc.list.gz
18. literature.list.gz
19. location.list.gz

20. miscellaneous.list.gz
21. miscellaneous-eompanies.list.gz
22. sound-mix.list.gz
23. soundtracks.list.gz
24. special-effects-companies.list.gz

5.2.1 Περιγραφή αρχείων actors.list και actresses.list

Τα δύο αρχεία περιέχουν πληροφορίες σχετικά με ηθοποιούς. Η δομή τους είναι η εξής:

```
ηθοποιός = ονοματεπώνυμο 1*(1*HTAB ρόλος LF) (LF / "")
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
ρόλος = έργο [1*SP χαρακτήρας] [1*SP κατάταξη]
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
χαρακτήρας = "[" *VCHAR "]"
κατάταξη = "<" 1*DIGIT ">"
```

Παρατηρήσεις:

- Όταν ο τίτλος του έργου είναι σε εισαγωγικά τότε πρόκειται για σειρά, επιπλέον αν ακολουθείται από το λεκτικό (mini) τότε αφορά μίνι σειρά.
- Το λεκτικό TV είναι ταινίες τηλεόρασης, ενώ το λεκτικό V είναι βιντεοκασέτες.

Και για τις δύο λίστες δημιουργήθηκε ο πίνακας actors, που αποδίδει ένα κλειδί σε κάθε ηθοποιό, και ο πίνακας movies2actors, που συνδέει τους ηθοποιούς με τις ταινίες, ως εξής:

```
CREATE TABLE `actors` (
  `actorid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  `sex` enum('M','F') DEFAULT NULL,
  `imdbid` mediumint(9) DEFAULT NULL,
  PRIMARY KEY (`actorid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=1736937 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2actors` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `actorid` mediumint(8) unsigned NOT NULL,
  `as_character` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `actorid` (`actorid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.2 Περιγραφή αρχείου cinematographers.list

Το αρχείο αυτό περιέχει του κινηματογραφιστές. Η δομή του είναι η εξής:

```
κινηματογραφιστής = ονοματεπώνυμο 1*(1*HTAB έργο LF) (LF / "")
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα παράχθηκε ο πίνακας cinematgrs και ο συνδετικός πίνακας movies2cinematgrs ως εξής:

```
CREATE TABLE `cinematgrs` (
  `cinematid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`cinematid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=92691 DEFAULT CHARSET=latin1;
CREATE TABLE `movies2cinematgrs` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `cinematid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `cinematid` (`cinematid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.3 Περιγραφή αρχείου color-info.list

Το αρχείο αυτό περιέχει πληροφορίες σχετικά με το χρώμα των έργων. Η δομή του είναι η εξής:

```
χρώμα = έργο 1*HTAB ("Color" / "Black and White") [1*HTAB σχόλιο] LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα παράχθηκε ο πίνακας colorinfo ως εξής:

```
CREATE TABLE `colorinfo` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `color` varchar(100) NOT NULL,
  KEY `movieid` (`movieid`),  KEY `color` (`color`(15))
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.4 Περιγραφή αρχείου composers.list

Το αρχείο αυτό περιέχει πληροφορίες σχετικά με μουσικοσυνθέτες. Η δομή του είναι η εξής:

```
μουσικοσυνθέτης = ονοματεπώνυμο 1*(1*HTAB έργο LF) (LF / "")
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας composers και ο συνδετικός πίνακας movies2composers ως εξής:

```
CREATE TABLE `composers` (
  `composerid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`composerid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=82269 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2composers` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `composerid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `composerid` (`composerid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.5 Περιγραφή αρχείου costume-designers.list

Το αρχείο αυτό περιέχει τους στυλίστες. Η δομή του είναι η εξής:

```
στυλίστας = ονοματεπώνυμο 1*(1*HTAB έργο LF) (LF / "")
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας costdesigners και ο συνδετικός πίνακας movies2costdes ως εξής:

```
CREATE TABLE `costdesigners` (
  `costdesid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`costdesid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=28691 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2costdes` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `costdesid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `costdesid` (`costdesid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.6 Περιγραφή αρχείου countries.list

Το αρχείο αυτό περιέχει τις χώρες των έργων. Η δομή του είναι η εξής:

```
χώρα = έργο 1*HTAB *VCHAR LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας countries και ο συνδετικός πίνακας movies2countries ως εξής:

```
CREATE TABLE `countries` (
  `countryid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `country` varchar(250) NOT NULL,
  PRIMARY KEY (`countryid`),
  KEY `name` (`country`(10))
) ENGINE=MyISAM AUTO_INCREMENT=204 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2countries` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `countryid` mediumint(8) unsigned NOT NULL,
  KEY `movieid` (`movieid`),
  KEY `countryid` (`countryid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.7 Περιγραφή αρχείου directors.list

Το αρχείο περιέχει τους σκηνοθέτες. Η δομή του είναι η εξής:

```
σκηνοθέτης = ονοματεπώνυμο 1*(1*HTAB έργο LF) LF
ονοματεπώνυμο = επώνυμο ",” SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας directors και ο συνδετικός πίνακας movies2directors ως εξής:

```
CREATE TABLE `directors` (
  `directorid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`directorid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=175603 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2directors` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `directorid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `directorid` (`directorid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.8 Περιγραφή αρχείου editors.list

Το αρχείο περιέχει τους διορθωτές. Η δομή του είναι η εξής:

```
διορθωτής = ονοματεπώνυμο 1*(1*HTAB έργο LF) LF
ονοματεπώνυμο = επώνυμο ",” SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας editors και ο συνδετικός πίνακας movies2editors ως εξής:

```
CREATE TABLE `editors` (
  `editorid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`editorid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=104211 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2editors` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `editorid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `editorid` (`editorid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.9 Περιγραφή αρχείου genres.list

Το αρχείο περιέχει τα είδη των έργων. Η δομή του είναι η εξής:

```
είδος = έργο 1*HTAB *VCHAR LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας genres ως εξής:

```
CREATE TABLE `genres` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `genre` varchar(50) NOT NULL,
  KEY `movieid` (`movieid`),
  KEY `genre` (`genre`(15))
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.10 Περιγραφή αρχείου keywords.list

Το αρχείο περιέχει τις λέξεις κλειδιά των έργων. Η δομή του είναι η εξής:

```
κλειδί = έργο 1*HTAB *VCHAR LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας keywords ως εξής:

```
CREATE TABLE `keywords` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `keyword` varchar(60) NOT NULL,
  KEY `movieid` (`movieid`),
  KEY `keyword` (`keyword`(15))
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
Περιγραφή αρχείου movies.list
Το αρχείο περιέχει τις παραγωγές. Η δομή του είναι η εξής:
παραγωγή = έργο 1*HTAB (έτος / "????") ["-" έτος / "????"] LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας movies ως εξής:

```
CREATE TABLE `movies` (
  `movieid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL,
  `year` varchar(100) DEFAULT NULL,
  `imdbid` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`movieid`),
  KEY `title` (`title`(15))
) ENGINE=MyISAM AUTO_INCREMENT=1464959 DEFAULT CHARSET=latin1;
```

5.2.11 Περιγραφή αρχείου mpaa-ratings-reasons.list

Το αρχείο περιέχει τις αξιολογήσεις. Η δομή του είναι η εξής:

```
αξιολόγηση = "MV: " έργο LF "RE: " *VCHAR 2LF 79("-") LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας mpaaratings ως εξής:

```
CREATE TABLE `mpaaratings` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `reasontext` text,
  KEY `movieid` (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.12 Περιγραφή αρχείου plot.list

Το αρχείο περιέχει τις υποθέσεις των έργων. Η δομή του είναι η εξής:

```
υπόθεση = "MV: " έργο 2LF 1*("PL: " *VCHAR LF) LF "BY: " *VCHAR 2LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας plots ως εξής:

```
CREATE TABLE `plots` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `plottext` text,
  PRIMARY KEY (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.13 Περιγραφή αρχείου producers.list

Το αρχείο περιέχει τους παραγωγούς. Η δομή του είναι η εξής:

```
παραγωγός = ονοματεπώνυμο 1*(1*HTAB έργο LF)
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας producers και ο συνδετικός πίνακας movies2producers ως εξής:

```
CREATE TABLE `producers` (
  `producerid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`producerid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=290521 DEFAULT CHARSET=latin1;
CREATE TABLE `movies2producers` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `producerid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `producerid` (`producerid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.14 Περιγραφή αρχείου production-companies.list

Το αρχείο περιέχει τις εταιρίες παραγωγής. Η δομή του είναι η εξής:

```
εταιρία = έργο 1*HTAB *VCHAR LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας prodcompanies και ο συνδετικός πίνακας movies2prodcompanies ως εξής:

```
CREATE TABLE `prodcompanies` (
  `prodcompanyid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`prodcompanyid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=157765 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2prodcompanies` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `prodcompanyid` mediumint(8) unsigned NOT NULL,
  KEY `movieid` (`movieid`),
  KEY `prodcompanyid` (`prodcompanyid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.15 Περιγραφή αρχείου production-designers.list

Το αρχείο περιέχει τους σχεδιαστές. Η δομή του είναι η εξής:

```
σχεδιαστής = ονοματεπώνυμο 1*(1*HTAB έργο LF) LF
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας proddesigners:

```
CREATE TABLE `proddesigners` (
  `proddesid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`proddesid`),
  KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=29946 DEFAULT CHARSET=latin1;
```

και ο συνδετικός πίνακας movies2proddes ως εξής:

```
CREATE TABLE `movies2proddes` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `proddesid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `proddesid` (`proddesid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.16 Περιγραφή αρχείου quotes.list

Το αρχείο περιέχει τις ατάκες των ταινιών. Η δομή του είναι η εξής:

```
ατάκα = έργο LF 1*(*VCHAR ": " *VCHAR 1*2LF) LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας quotes ως εξής:

```
CREATE TABLE `quotes` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `quotetext` mediumtext,
  KEY `movieid` (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.17 Περιγραφή αρχείου ratings.list

Το αρχείο περιέχει τις βαθμολογίες των ταινιών. Η δομή του είναι η εξής:

```
βαθμολογία = 6SP κατανομή πλήθος βαθμός 2SP έργο LF
κατανομή = 10("." / DIGIT / "*")
πλήθος = 8(SP / DIGIT)
βαθμός = 4(SP / DIGIT) "." DIGIT
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας ratings ως εξής:

```
CREATE TABLE `ratings` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `rank` char(4) NOT NULL,
  `votes` mediumint(8) unsigned DEFAULT NULL,
  `distribution` char(10) NOT NULL,
  KEY `movieid` (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.18 Περιγραφή αρχείου release-dates.list

Το αρχείο περιέχει τις πρεμιέρες των ταινιών. Η δομή του είναι η εξής:

```
πρεμιέρα = έργο 1*(1*HTAB) *VCHAR ":" ημερομηνία LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
ημερομηνία = 1*2DIGIT SP ("January" / "February" / "March" / "April" /
"May" / "June" / "July" / "August" / "September" / "October" /
"November" / "December") SP έτος
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας releasedates ως εξής:

```
CREATE TABLE `releasedates` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `releasedate` int(11) DEFAULT NULL,
  KEY `movieid` (`movieid`),
  KEY `releasedate` (`releasedate`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.19 Περιγραφή αρχείου running-times.list

Το αρχείο περιέχει τις διάρκειες των ταινιών. Η δομή του είναι η εξής:

```
διάρκεια = έργο 1*HTAB [*VCHAR ":"] 1*DIGIT [1*HTAB σχόλιο] LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE [" (mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας runningtimes και ο συνδετικός πίνακας movies2runningtimes ως εξής:

```
CREATE TABLE `runningtimes` (
  `runningtimeid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `runningtime` varchar(250) NOT NULL,
  PRIMARY KEY (`runningtimeid`),
  KEY `runningtime` (`runningtime`(10))
) ENGINE=MyISAM AUTO_INCREMENT=10385 DEFAULT CHARSET=latin1;

CREATE TABLE `movies2runningtimes` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `runningtimeid` mediumint(8) unsigned NOT NULL,
  KEY `movieid` (`movieid`),
  KEY `runningtimeid` (`runningtimeid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.20 Περιγραφή αρχείου taglines.list

Το αρχείο περιέχει τα σλόγκαν των ταινιών. Η δομή του είναι η εξής:

```
σλόγκαν = "# " έργο LF 1*(HTAB *VCHAR LF) LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας taglines ως εξής:

```
CREATE TABLE `taglines` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `taglinetext` text,
  KEY `movieid` (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.21 Περιγραφή αρχείου trivia.list

Το αρχείο περιέχει γεγονότα των ταινιών. Η δομή του είναι η εξής:

```
γεγονός = "# " έργο LF 1*("- " 1*(*VCHAR LF) LF) LF
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας trivia ως εξής:

```
CREATE TABLE `trivia` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `triviatext` text,
  KEY `movieid` (`movieid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.2.22 Περιγραφή αρχείου writers.list

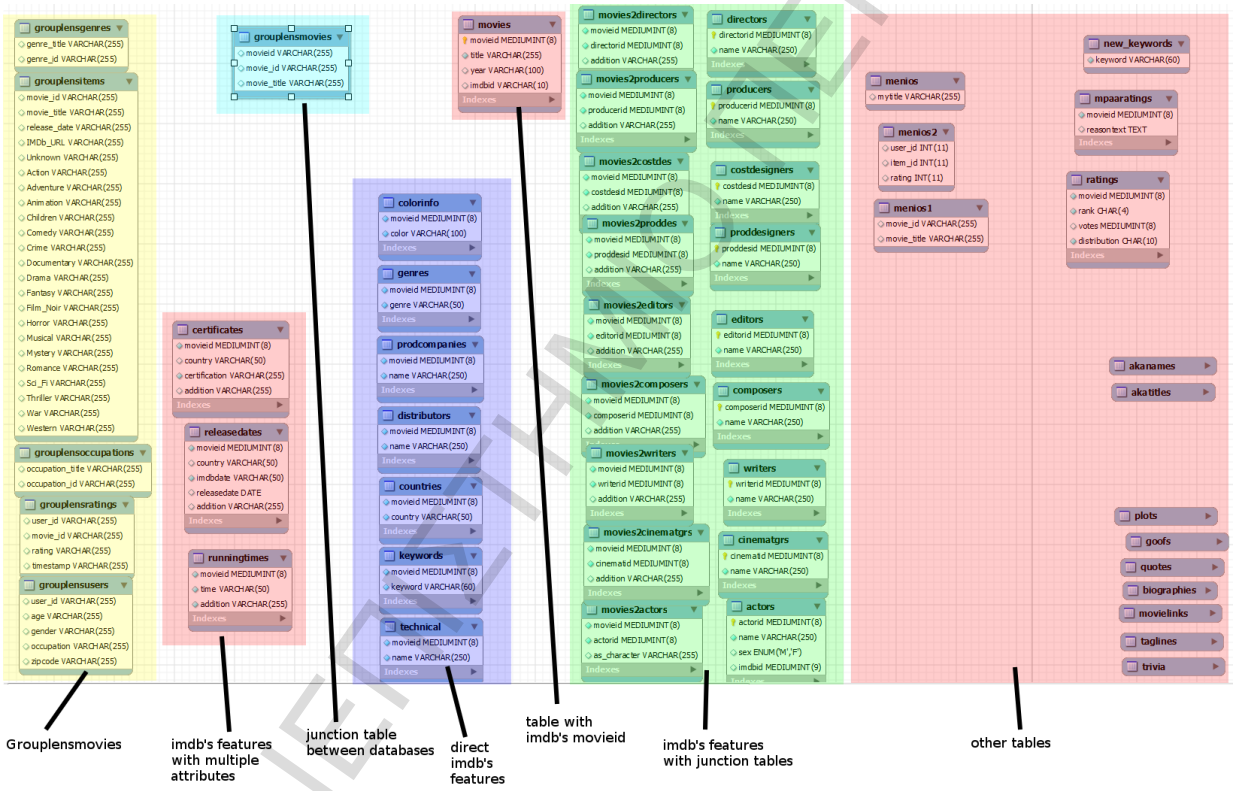
Το αρχείο περιέχει τους σεναριογράφους. Η δομή του είναι η εξής:

```
σεναριογράφος = ονοματεπώνυμο 1*(1*HTAB έργο [1*SP αλληλουχία] LF)
ονοματεπώνυμο = επώνυμο "," SP όνομα
επώνυμο = *VCHAR
όνομα = *VCHAR / (ALPHA ".")
έργο = (σειρά / ταινία) *(1*SP σχόλιο)
σειρά = DQUOTE τίτλος DQUOTE ["(mini)"] SP έτος SP επεισόδιο
επεισόδιο = "{" τίτλος SP "(#" 1*DIGIT "." 1*DIGIT ")}"
ταινία = τίτλος SP "(" έτος ")" SP ["(TV)" / "(V)"]
τίτλος = *VCHAR
έτος = 4DIGIT
σχόλιο = "(" *VCHAR ")"
αλληλουχία = "<" DIGIT "," DIGIT "," DIGIT ">"
```

Για αυτήν τη λίστα δημιουργήθηκε ο πίνακας writers και ο συνδετικός πίνακας movies2writers ως εξής:

```
CREATE TABLE `writers` (
  `writerid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(250) NOT NULL,
  PRIMARY KEY (`writerid`), KEY `name` (`name`(10))
) ENGINE=MyISAM AUTO_INCREMENT=251912 DEFAULT CHARSET=latin1;
CREATE TABLE `movies2writers` (
  `movieid` mediumint(8) unsigned NOT NULL,
  `writerid` mediumint(8) unsigned NOT NULL,
  `addition` varchar(255) DEFAULT NULL,
  KEY `movieid` (`movieid`), KEY `writerid` (`writerid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

5.3 Συσχέτιση βάσεων



Εικόνα 3: Εικόνα με όλους τους πίνακες του συστήματος.

Για τη συσχέτιση των δύο βάσεων αντιγράφηκαν οι παραπάνω πίνακες σε μία βάση δεδομένων και χρειάστηκε να παραχθεί ένας συνδετικός πίνακας, τέτοιος ώστε κάθε πλειάδα του να περιέχει το κλειδί μίας ταινίας από τη MovieLens και το αντίστοιχο κλειδί της ίδιας ταινίας από τον πίνακα movies της IMDb. Όμως τα κλειδιά των ταινιών δεν είναι ίδια στις δύο βάσεις, γι' αυτό χρειάστηκε ένας τρόπος να συγκριθούν οι ταινίες μεταξύ τους. Ο μόνος ίσως συνδετικός κρίκος είναι οι τίτλοι των ταινιών, οι οποίοι δυστυχώς δεν ταιριάζουν πάντα αφού υπάρχουν διαφόρων ειδών ανομοιομορφίες στη σύνταξή τους [15].

Συγκεκριμένα παρατηρήθηκε ότι η MovieLens εμφανίζει τα άρθρα των τίτλων σε διαφορετική θέση, π.χ. η ταινία "The Matrix (1999)" εμφανίζεται ως "Matrix, The (1999)". Γι' αυτόν τον λόγο πραγματοποιήθηκε η παρακάτω αντιγραφή σε μία προσωρινή βάση αντικαθιστώντας τους τίτλους της MovieLens που έχουν μετατοπισμένα άρθρα:

```
INSERT INTO temp.altered_movies
SELECT recommender.ml_movies.MovieId,
      CONCAT('The ', REPLACE(recommender.ml_movies.MovieTitle,',
The',''))
FROM recommender.ml_movies
WHERE recommender.ml_movies.MovieTitle LIKE '%, The%'
UNION
SELECT recommender.ml_movies.MovieId,
      CONCAT('An ', REPLACE(recommender.ml_movies.MovieTitle, ', An',
''))
FROM recommender.ml_movies
WHERE recommender.ml_movies.MovieTitle LIKE '%, An%'
UNION
SELECT recommender.ml_movies.MovieId,
      CONCAT('A ', REPLACE(recommender.ml_movies.MovieTitle, ', A ', '
'))
FROM recommender.ml_movies
WHERE recommender.ml_movies.MovieTitle LIKE '%, A %';
```

Στη συνέχεια πραγματοποιήθηκε η συσχέτιση των βάσεων με την ακόλουθη εντολή:

```
INSERT INTO recommender.ml2imdb (recommender.ml2imdb.ml_id,
recommender.ml2imdb.imdb_id)
SELECT temp.altered_movies.id, recommender.imdb_movies.movieid
FROM temp.altered_movies, recommender.imdb_movies
WHERE UPPER(temp.altered_movies.title) COLLATE utf8_bin =
UPPER(recommender.imdb_movies.title);
```

η οποία γεμίζει τον πίνακα $ml2imdb$ με τις κοινές ταινίες των δύο βάσεων δεδομένων, δηλαδή αυτές που έχουν κοινό τίτλο αφού πρώτα έχει προηγηθεί η μεταφορά των άρθρων The, An, και A στην κατάλληλη θέση και χωρίς να λαμβάνονται υπόψη τόνοι ή αν τα γράμματα είναι κεφαλαία ή μικρά.

6 Εκτέλεση πειράματος

Από τη διαδικασία εξαγωγής δεδομένων και σύνδεσης των δύο βάσεων δεδομένων παράχθηκε πίνακας βαθμολογιών για 2820 ταινίες από 6040 χρήστες χρησιμοποιώντας την εξής εντολή SQL:

```
SELECT distinct id
FROM documents LEFT JOIN ml_ratings
ON id=movieid
WHERE keywords IS NOT NULL
AND plots IS NOT NULL
AND rating IS NOT NULL;
```

Όπως ήταν αναμενόμενο, από τις 17.032.800 θέσεις μόνο οι 879.227 περιείχαν τιμές. Δηλαδή, παρουσιάζεται σαφώς πρόβλημα αραιών δεδομένων, αφού μόλις το 5% του πίνακα βαθμολογιών περιέχει τιμές. Υπό αυτές τις συνθήκες πραγματοποιήθηκε έρευνα και επιλέχθηκε μία πρόταση από τη βιβλιογραφία που να αντιμετωπίζει το πρόβλημα υλοποιώντας ένα σύστημα συστάσεων.

Για τη διεξαγωγή του πειράματος υλοποιήθηκε σε MATLAB η μέθοδος που περιγράφεται σε δημοσίευση των Sarwar et al. [16], κατά την οποία αξιοποιούνται μόνο οι βαθμολογίες των χρηστών για κάθε ταινία. Όπως είναι προφανές πρόκειται για μία τεχνική συνεργατικού φιλτραρίσματος σε αντίθεση με την προτεινόμενη που βασίζεται στα στοιχεία, αφού χρησιμοποιεί τα χαρακτηριστικά των ταινιών.

Για την ελαχιστοποίηση της επιρροής των αποτελεσμάτων από τα δεδομένα, πρέπει να πραγματοποιηθεί τυχαία κατάτμηση των βαθμολογίων σε πολλαπλά μέρη ώστε να γίνουν επαναλαμβανόμενες δοκιμές της μεθόδου με διαφορετικά δεδομένα. Σε κάθε δοκιμή πρέπει να αποκρύπτεται το εκάστοτε τμήμα, με σκοπό να προσεγγιστούν οι βαθμολογίες αξιοποιώντας το υπόλοιπο μέρος των βαθμολογιών που παραμένει φανερό.

Ως μετρική χρησιμοποιείται το Μέσο Απόλυτο Σφάλμα (Mean Absolute Error) που υπολογίζεται λαμβάνοντας τον μέσο όρο της απόλυτης διαφοράς των βαθμολογιών που προσεγγίζονται από τη μέθοδο, $f(u,i)$, από τις βαθμολογίες, r_{ui}, του εκάστοτε δοκιμαστικού τμήματος, $test$, ως εξής:

$$MAE(f) = \frac{1}{|\text{test}|} \sum_{r_{ui} \in \text{test}} |f(u,i) - r_{ui}|$$

6.1 Υλοποίηση

Για την υλοποίηση της μεθόδου γράφτηκε κώδικας σε MATLAB, ο οποίος δέχεται ως παραμέτρους τον πίνακα με τις βαθμολογίες, $train$, και το μειωμένο πλήθος διαστάσεων, k, που θα προκύψει με τη μέθοδο SVD, η οποία περιγράφεται αναλυτικά σε επόμενο κεφάλαιο.

Ο πίνακας train περιέχει μηδενικά στη θέση των βαθμολογιών που δεν είναι γνωστές. Σύμφωνα με τη μέθοδο των Sarvar et al. προτείνεται να κανονικοποιηθούν οι βαθμολογίες, αφού πρώτα συμπληρωθεί η μέση βαθμολογία κάθε ταινίας στη θέση των μηδενικών, αφαιρώντας τη μέση βαθμολογία κάθε χρήστη από κάθε βαθμολογία του.

Στη συνέχεια πραγματοποιείται μείωση διάστασης σε πλήθος k, χρησιμοποιώντας την τεχνική SVD, με την οποία παράγονται τρεις πίνακες, U, S και V. Για την πρόβλεψη της βαθμολογίας κάθε χρήστη, u, για κάθε ταινία, i, υπολογίζονται οι πίνακες $Q = \sqrt{S} \cdot V'$ και $P = U \cdot \sqrt{S}$ και προστίθεται η μέση τιμή των βαθμολογιών του χρήστη με το εσωτερικό γινόμενο $Q_i^T P_u$.

Ο κώδικας που ακολουθεί υλοποιεί τα βήματα που περιγράφονται παραπάνω, ώστε αφού καλύψει τα κενά με μέσες τιμές, κανονικοποιήσει όλες τις βαθμολογίες και ελαττώσει τη διάσταση, παράγει τις εκτιμήσεις του συστήματος για τις βαθμολογίες που είναι πιθανό να αντιστοιχούσαν οι χρήστες σε κάθε ταινία.

```
function test = predict(train,k)
    average_I = sum(train,1)./sum(train~=0,1);
    average_I(~isfinite(average_I))=0;
    average_I(isnan(average_I))=0;
    average_U = sum(train,2)./sum(train~=0,2);
    average_U(~isfinite(average_U))=0;
    average_U(isnan(average_U))=0;
    [m, n]=size(train);
    for u = 1 : m
        for i = 1 : n
            if train(u,i)==0
                train(u,i) = average_I(i);
            end
            train(u,i) = train(u,i) - average_U(u);
        end
    end
    [U,S,V] = svds(train,k);
    sqrt_S = S.^(1/2);
    P=U*sqrt_S;
    Q=sqrt_S*V';
    test=zeros(m,n);
    for u = 1:m
        for i = 1:n
            test(u,i)=average_U(u)+dot(P(u,:),Q(:,i));
        end
    end
end
```

6.2 Κατάτμηση πίνακα βαθμολογιών

Για την κατάτμηση γράφτηκε κώδικας σε MATLAB, ο οποίος δέχεται ως παραμέτρους τον πίνακα των βαθμολογιών, $matrix$, και το πλήθος των κατατμήσεων, k.

Η λογική που ακολουθείται βασίζεται στην τοποθέτηση των βαθμολογίων σε μία τυχαία μετάθεση (permutation) και τον διαμερισμό της σε k ίσα μέρη. Να σημειωθεί ότι στην περίπτωση όπου δε διαμερίζεται ακριβώς το σύνολο των βαθμολογιών, το υπολειπόμενο υποσύνολο απλά αγνοείται.

Στον κώδικα που αναπτύχθηκε γι' αυτόν το σκοπό υπολογίζεται το μέγεθος της κάθε κατάτμησης διαιρώντας το πλήθος των μη μηδενικών τιμών δια k και στρογγυλοποιώντας το αποτέλεσμα προς τα κάτω. Το πλήθος των μη μηδενικών τιμών μπορεί να βρεθεί αξιοποιώντας την εντολή find, η οποία αναλαμβάνει να εξάγει τις μη μηδενικές βαθμολογίες από τον πίνακα εντοπίζοντας παράλληλα τους δείκτες κάθε τιμής. Είναι προφανές ότι το πλήθος των δεικτών είναι ίσο με το πλήθος των μη μηδενικών τιμών.

Η τυχαία μετάθεση προκύπτει από τη χρήση της εντολής randperm και δίνοντας ως παράμετρο το πλήθος των στοιχείων. Στη συνέχεια σε κάθε κατάτμηση i αντιστοιχίζονται οι βαθμολογίες των επόμενων τυχαίων θέσεων και τοποθετούνται σε έναν πίνακα επιβεβαίωσης των αποτελεσμάτων.

Παράλληλα, αφαιρείται κάθε κατάτμηση από τον αντίστοιχο πίνακα βαθμολογιών αντικαθιστώντας τις τιμές με μηδενικά. Εκ των πραγμάτων δημιουργούνται k νέοι πίνακες βαθμολογιών, $train$, με σκοπό να προσεγγιστούν οι αφαιρεμένες βαθμολογίες, οι οποίες κρατούνται σε αντίστοιχους k πίνακες επιβεβαίωσης, $validate$, για να ελεγχθεί η επιτυχία της μεθόδου.

Χρησιμοποιώντας τον κώδικα που ακολουθεί, για λόγους ευκολίας, αντί να παραχθούν k ξεχωριστοί πίνακες, επιλέχθηκε να δημιουργηθεί τρισδιάστατος πίνακας, που η πρώτη του

διάσταση είναι μεγέθους k, ο οποίος έπειτα αποθηκεύτηκε σε αρχείο MATLAB με όνομα $trainset.mat$. Αντίστοιχα, για τους πίνακες επιβεβαίωσης δημιουργήθηκε το αρχείο $validate.mat$.

```
function [train, validate] = fragment(matrix,k)
    [I, J, V] = find(matrix);
    P = randperm(length(I));
    [m,n]=size(matrix);
    train = zeros(k,m,n);
    validate = zeros(k,m,n);
    l = floor(length(I)/k);
    for i = 1:k
        train(i,:,:) = matrix(:,:);
        for j = (i-1)*l+1 : i*l
            p = P(j);
            validate(i,I(p),J(p)) = V(p);
            train(i,I(p),J(p)) = 0;
        end
    end
end
```

6.3 Αποτελέσματα

Με τη χρήση των παραπάνω μεθόδων εκτελέστηκαν 10 πειράματα στις 10 κατατμήσεις που αποθηκεύτηκαν στο αρχείο $trainset.mat$. Για κάθε κατάτμηση υπολογίστηκε η μέση, η ελάχιστη και η μέγιστη απόλυτη διαφορά μεταξύ της πραγματικής βαθμολογίας και της εκτίμησής της από το σύστημα συστάσεων. Για το σκοπό αυτό γράφηκε ο εξής κώδικας σε MATLAB:

```
function [low, average, high] =
      evaluate(trainsetname,validsetname,p,k)
    trainset=importdata(trainsetname);
    train(:,:)=trainset(p,:,:);
    clear(trainsetname);
    test = predict(train,k);
    validset=importdata(validsetname);
    valid(:,:)=validset(p,:,:);
    clear(validsetname);
    distances = abs(test(valid~=0)-valid(valid~=0));
    low = min(distances);
    average = mean(distances);
    high = max(distances);
end
```

Για τον υπολογισμό αυτό εκτελέστηκαν οι εξής εντολές σε MATLAB:

- Κατάτμηση 1

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',1,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7297
min: 6.3345e-06
max: 4.9656
```

- Κατάτμηση 2

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',2,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7310
min: 6.7324e-05
max: 4.8963
```

- Κατάτμηση 3

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',3,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7300
min: 7.4686e-06
max: 5.0020
```

- Κατάτμηση 4

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',4,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7295
min: 6.8753e-06
max: 5.0555
```

- Κατάτμηση 5

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',5,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7328
min: 1.6325e-05
max: 5.1179
```

- Κατάτμηση 6

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',6,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7272
min: 2.9114e-06
max: 4.2145
```

- Κατάτμηση 7

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',7,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7313
min: 5.5048e-06
max: 5.0054
```

- Κατάτμηση 8

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',8,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7323
min: 5.1748e-06
max: 4.0331
```

- Κατάτμηση 9

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',9,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7309
min: 1.8160e-05
max: 4.1638
```

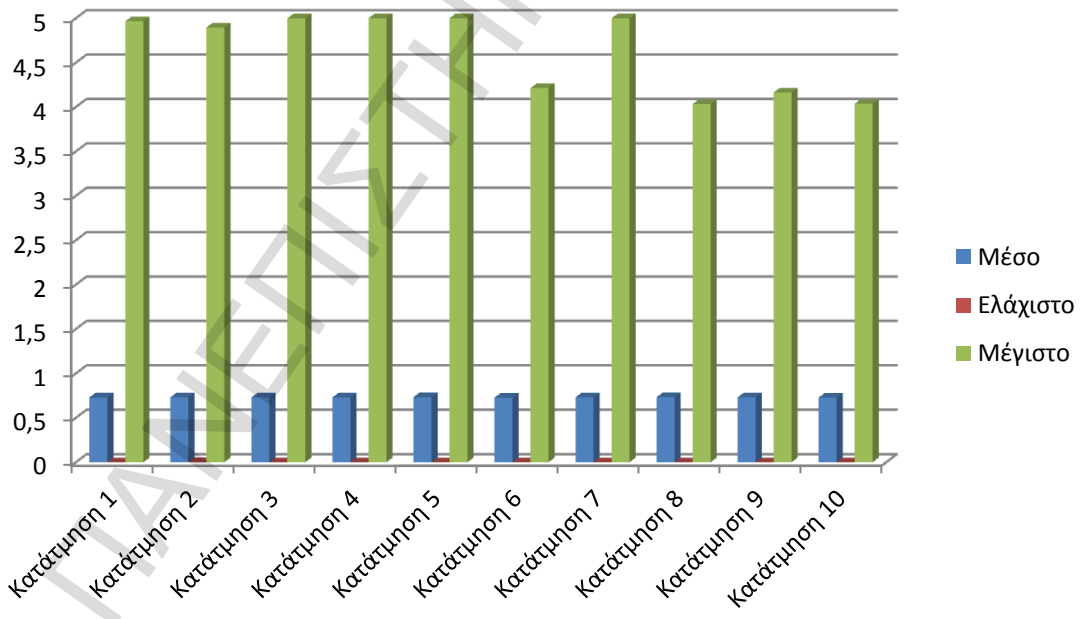
- Κατάτμηση 10

```
[min,mean,max] = evaluate('trainset.mat','validset.mat',10,10);
```

Με τα ακόλουθα αποτελέσματα:

```
mean: 0.7282
min: 3.8684e-05
max: 4.0365
```

Από τα παραπάνω δεδομένα προκύπτει το εξής διάγραμμα:

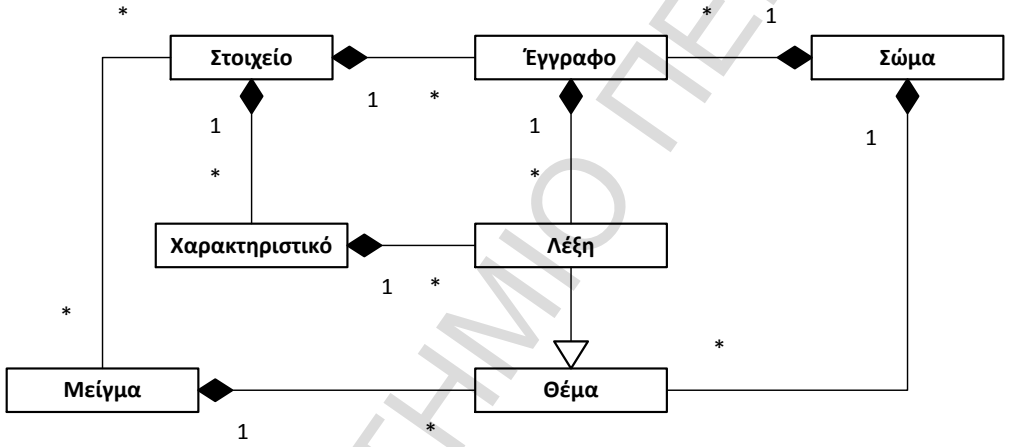


από το οποίο διαφαίνεται ότι παρόλο που σε όλες τις κατατμήσεις η απόλυτη μέση διαφορά δεν ξεπερνάει το 1 αστέρι βαθμολογίας (με μέγιστο τα 5), 6 στις 10 περιπτώσεις υπάρχει έστω και μία βαθμολογία που αποκλίνει κατά 5 αστέρια, και στις υπόλοιπες 4 κατά 4 αστέρια. Ας σημειωθεί βέβαια πως σε όλες τις περιπτώσεις υπήρχαν βαθμολογίες που η απόλυτη διαφορά ήταν αμελητέα.

7 Περιγραφή προτεινόμενης μεθόδου

Στην παρούσα εργασία γίνεται μία προσπάθεια για την αναπαράσταση των στοιχείων υπό μορφή θεμάτων (topics). Υπό μία έννοια γίνεται χρήση συστήματος συστάσεων βάσει περιεχομένου, αφού σε πρώτη φάση θα παρουσιάζονται στον χρήστη στοιχεία που έχουν ίδιο θέμα με αυτά που είχε προτιμήσει στο παρελθόν.

Το πρόβλημα με τα συστήματα συστάσεων βάσει περιεχομένου είναι ότι τα χαρακτηριστικά των στοιχείων μπορεί να είναι πολλά και κατά συνέπεια τα διανύσματα χαρακτηριστικών τους αρκετά αραιά, με αποτέλεσμα να είναι απίθανη η εύρεση ομοιοτήτων μεταξύ των στοιχείων. Με τη δημιουργία εγγράφων από τα χαρακτηριστικά κάθε στοιχείου είναι δυνατή η παραγωγή και μετέπειτα η σύγκριση των θεμάτων των στοιχείων για την εύρεση ομοιοτήτων. Για την επιλογή των θεμάτων, που δύναται να απαρτίσουν κάθε στοιχείο, πρέπει να συλλεχθούν όλα τα παραχθέντα έγγραφα σε ένα σώμα (corpus) από το οποίο θα προκύψουν τα καταλληλότερα θέματα χρησιμοποιώντας αλγόριθμους μοντελοποίησης θεμάτων (topic modeling), όπως ο LDA. Η συνεισφορά των επιλεχθέντων θεμάτων σε κάθε στοιχείο αναπαρίσταται με ένα μείγμα προσδιορίζοντας το στοιχείο αυτό με ένα σχετικά πυκνό διάνυσμα. Στην παρακάτω εικόνα φαίνεται με γραφικό τρόπο η συσχέτιση των εννοιών.



Εικόνα 4: Διάγραμμα στατικής δομής της μεθόδου. Για κάθε στοιχείο παράγεται ένα πλήθος εγγράφων, αντιστοιχίζοντας κάθε χαρακτηριστικό του στοιχείου σε μία ή περισσότερες λέξεις. Έπειτα τα έγγραφα συνιστούν ένα σώμα στο οποίο εντοπίζονται τα διάφορα θέματα. Τέλος, συσχετίζεται ένα μείγμα από σταθερό αριθμό θεμάτων με κάθε στοιχείο.

7.1 Μοντελοποίηση θεμάτων

Τα μοντέλα θεμάτων βασίζονται στην ιδέα ότι τα έγγραφα αποτελούνται από μείγματα θεμάτων, όπου ένα θέμα είναι η κατανομή πιθανότητας των λέξεων. Ένα μοντέλο θέματος είναι ένα γεννητικό μοντέλο για έγγραφα, δηλαδή περιγράφει μία απλή πιθανολογική διαδικασία με την οποία μπορούν να δημιουργηθούν έγγραφα. Για τη δημιουργία ενός νέου εγγράφου αρχικά επιλέγεται μία κατανομή θεμάτων. Έπειτα, κάθε λέξη σε αυτό το έγγραφο προκύπτει από ένα τυχαίο θέμα σύμφωνα με την κατανομή. Χρησιμοποιώντας καθιερωμένες τεχνικές στατιστικής η διαδικασία μπορεί να αντιστραφεί, ώστε να προσδιοριστεί το σύνολο των θεμάτων βάσει του οποίου δημιουργήθηκε μία συλλογή εγγράφων [9].

7.2 Latent Dirichlet allocation

Η τεχνική latent Dirichlet allocation (LDA) [10] μοντελοποιεί συλλογές διακριτών δεδομένων, όπως σώματα κειμένων. Είναι ένα ιεραρχικό Μπεϋζιανό μοντέλο τριών επιπέδων, στο οποίο κάθε αντικείμενο μίας συλλογής μοντελοποιείται ως πεπερασμένο μείγμα από ένα σύνολο θεμάτων. Με τη σειρά του, κάθε θέμα μοντελοποιείται ως ένα άπειρο μείγμα από ένα σύνολο πιθανοτήτων επί των θεμάτων.

Παρακάτω, κατά τον ορισμό των εννοιών, χρησιμοποιούνται οι όροι λέξη, έγγραφο και σώμα, παρόλο που η LDA μπορεί να λειτουργήσει και με διαφορετικού είδους δεδομένα, που δεν είναι κείμενα. Αυτό συμβαίνει επειδή οι συγκεκριμένες έννοιες προσφέρονται καλύτερα για την κατανόηση πιο αφηρημένων ιδεών, όπως αυτή του θέματος.

- Μία **λέξη** (word) είναι η βασική μονάδα των διακριτών δεδομένων, που ορίζεται ως ένα στοιχείο ενός λεξιλογίου δεικτοδοτούμενο από $\{1, \dots, V\}$. Η αναπαράσταση των λέξεων γίνεται από μοναδιαία διανύσματα μεγέθους V, που έχουν μία συνιστώσα να ισούται με το 1 και τις υπόλοιπες με το 0.
- Ένα **έγγραφο** (document) είναι η σειρά από N λέξεις, που συμβολίζεται με $\boldsymbol{w} = (w_1, w_2, \dots, w_N)$, όπου w_n είναι η n-ιοστή λέξη της σειράς.
- Ένα **σώμα** (corpus) είναι το σύνολο M εγγράφων, που συμβολίζονται με $D = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_M\}$.

Η βασική ιδέα της τεχνικής είναι ότι τα έγγραφα παριστάνονται ως τυχαία μείγματα από λανθάνοντα θέματα, όπου κάθε θέμα χαρακτηρίζεται από μία κατανομή λέξεων. Η LDA υποθέτει ότι τα έγγραφα παράγονται από μία γεννητική διαδικασία, κατά την οποία ορίζεται μία κοινή κατανομή πιθανότητας μεταξύ των ορατών και κρυφών τυχαίων μεταβλητών. Οι εμφανείς μεταβλητές αφορούν τις λέξεις, ενώ οι κρυφές σχετίζονται με τα θέματα, την κατανομή των θεμάτων ανά έγγραφο και την κατανομή των θεμάτων ανά λέξη κάθε εγγράφου.

Το βασικό υπολογιστικό πρόβλημα της LDA είναι να υπονοηθούν οι κρυφές μεταβλητές ενός σώματος κειμένων αξιοποιώντας τις ορατές μεταβλητές. Ουσιαστικά πρόκειται για την αντιστροφή της γεννητικής διαδικασίας με την οποία πιθανολογείται η παραγωγή των εγγράφων [11]. Για την παραγωγή εγγράφων ακολουθείται η παρακάτω διαδικασία για κάθε έγγραφο $\boldsymbol{w}$ σε ένα σώμα D:

1. Επιλογή N ~ Poisson(ξ)
2. Επιλογή θ ~ Dir(α)
3. Για κάθε μία από τις N λέξεις w_n:
    a. Επιλογή ενός θέματος z_n~ Multinomial(θ)
    b. Επιλογή μίας λέξης w_n από $p(w_n|z_n, \beta)$, μία πολυωνυμική πιθανότητα εξαρτώμενη από το θέμα z_n.

Το τυχαίο διάνυσμα θ ακολουθεί κατανομή Dirichlet, δεδομένου ενός διανύσματος α, όπου και τα δύο διανύσματα είναι διάστασης k. Η συγκεκριμένη κατανομή έχει την ακόλουθη πυκνότητα πιθανότητας:

$$p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(a_i)} \theta_1^{\alpha_1 - 1} \dots \theta_k^{\alpha_k - 1}$$

όπου με $\Gamma(x)$ συμβολίζεται η συνάρτηση Γάμμα.

Δεδομένων των παραμέτρων α και β, η κοινή κατανομή ενός μείγματος θεμάτων θ, ενός συνόλου από N θέματα $\boldsymbol{z}$, και ενός συνόλου από N λέξεις $\boldsymbol{w}$ δίνεται από την ακόλουθη σχέση:

$$p(\theta, \boldsymbol{z}, \boldsymbol{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

όπου το $p(z_n|\theta)$ είναι απλά το θ_i για κάθε i τέτοιο ώστε το $z_n^i = 1$.

Η περιθώρια κατανομή ενός εγγράφου υπολογίζεται με το ολοκλήρωμα ως προς θ του αθροίσματος ως προς z ως εξής:

$$p(\boldsymbol{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta$$

Με το γινόμενο των οριακών πιθανοτήτων των εγγράφων λαμβάνεται η πιθανότητα ενός σώματος:

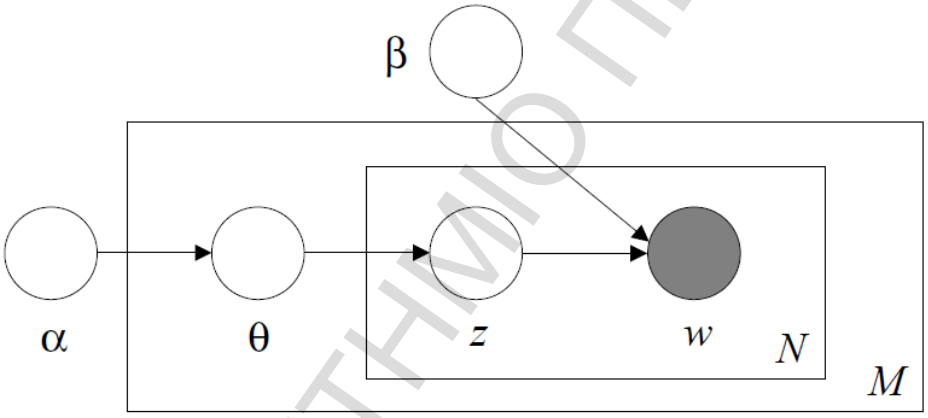
$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn},\beta) \right) d\theta_d$$

Τέλος, η γεννητική διαδικασία της LDA αντιστοιχεί στην εξής κοινή κατανομή των ορατών και κρυφών μεταβλητών:

$$p(\beta,\theta,\boldsymbol{z},\boldsymbol{w}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta, z_{d,n}) \right)$$

στην οποία παρατηρείται ένα πλήθος εξαρτήσεων. Για παράδειγμα η ανάθεση θέματος $z_{d,n}$ εξαρτάται από τη θ_d αναλογία θέματος ανά έγγραφο. Ομοίως η ορατή λέξη $w_{d,n}$ εξαρτάται από την ανάθεση θέματος $z_{d,n}$ και από όλα τα θέματα β.

Αυτές οι εξαρτήσεις, που ορίζουν την LDA, κωδικοποιούνται στις στατιστικές παραδοχές που βρίσκονται πίσω από τη γεννητική διαδικασία, υπό τη μαθηματική μορφή της κοινής κατανομής. Οι εξαρτήσεις αυτές, καθώς και τα 3 επίπεδα της τεχνικής, φαίνονται στην παρακάτω εικόνα με γραφικό τρόπο. Οι μεταβλητές α, β ορίζονται στο επίπεδο του σώματος, η μεταβλητή θ αφορά το επίπεδο του εγγράφου, ενώ η μεταβλητές z και w σχετίζονται με την εκάστοτε λέξη.



Εικόνα 5: Γραφικό μοντέλο που αναπαριστά την τεχνική LDA. Τα ορθογώνια παραλληλόγραμμα είναι «πλάκες» που συνιστούν επαναλήψεις. Η εξωτερική πλάκα αναπαριστά έγγραφα, ενώ η εσωτερική αναπαριστά την επαναλαμβανόμενη επιλογή θεμάτων και λέξεων εντός του εγγράφου.

Αφού ορίστηκαν η γεννητική διαδικασία της LDA και οι εξαρτήσεις που προκύπτουν, πρέπει να περιγραφεί και το υπολογιστικό πρόβλημα κατά τη λύση του οποίου υπολογίζεται η κατανομή της δομής των θεμάτων, τα οποία είναι κρυφά στοιχεία, δεδομένων των εγγράφων, τα οποία είναι ορατά. Δηλαδή:

$$p(\beta,\theta,\boldsymbol{z}|\boldsymbol{w}) = \frac{p(\beta,\theta,\boldsymbol{z},\boldsymbol{w})}{p(\boldsymbol{w})}$$

όπου ο αριθμητής είναι η κοινή κατανομή όλων των τυχαίων μεταβλητών, η οποία μπορεί εύκολα να υπολογιστεί. Ο παρονομαστής είναι η περιθώρια πιθανότητα των ορατών μεταβλητών, δηλαδή η πιθανότητα παρατήρησης του ορατού σώματος κειμένων υπό οποιουδήποτε μοντέλου θεμάτων. Θεωρητικά μπορεί να υπολογιστεί αθροίζοντας τις κοινές κατανομές όλων των πιθανών παραλλαγών των κρυφών δομών των θεμάτων.

Το πλήθος όλων των πιθανών δομών των θεμάτων, όμως, αυξάνει εκθετικά, δυσχεραίνοντας τη λύση του προβλήματος. Οι αλγόριθμοι μοντελοποίησης θεμάτων προσεγγίζουν την παραπάνω εξίσωση δομώντας μία εναλλακτική κατανομή της κρυφή δομής θεμάτων, η οποία προσαρμόζεται κοντά στην πραγματική λύση. Γενικά οι αλγόριθμοι αυτοί

κατατάσσονται σε δύο κατηγορίες: α) στους δειγματοληπτικούς (sampling) και β) στους μεταβολικούς (variational).

Οι δειγματοληπτικοί αλγόριθμοι προσπαθούν να συλλέξουν δείγματα από την πραγματική λύση ώστε να την προσεγγίσουν με μία εμπειρική κατανομή. Ο πιο συχνός χρησιμοποιούμενος δειγματοληπτικός αλγόριθμος μοντελοποίησης θεμάτων είναι ο Gibbs sampling, κατά τον οποίο κατασκευάζεται μία Μαρκοβιανή αλυσίδα, δηλαδή μία αλληλουχία τυχαίων μεταβλητών, όπου κάθε μία προκύπτει από την προηγούμενή της. Η οριακή αλυσίδα ορίζεται επί των κρυφών στοιχείων για ένα συγκεκριμένο σώμα κειμένων και ο αλγόριθμος πρέπει να διατρέξει την αλυσίδα για αρκετό χρόνο συλλέγοντας δείγματα της οριακής κατανομής, η οποία αποτελεί και τη λύση του προβλήματος.

Οι μεταβολικοί αλγόριθμοι αποτελούν ντετερμινιστικές εναλλακτικές μεθόδους έναντι των δειγματοληπτικών. Αντί να προσεγγίζουν τη λύση με δείγματα, οι μεταβολικές μέθοδοι διερευνούν μία παραμετρική οικογένεια κατανομών των κρυφών δομών, ώστε να βρουν το μέλος της οικογένειας που είναι πιο κοντά στην πραγματική λύση. Με λίγα λόγια, το υπολογιστικό πρόβλημα μετατρέπεται σε πρόβλημα βελτιστοποίησης.

Σε γενικές γραμμές, και οι δύο τύποι αλγορίθμων αναζητούν τις δομές των θεμάτων. Η συλλογή των εγγράφων, που είναι και τα ορατά στοιχεία του προβλήματος, παραμένουν σταθερά και λειτουργούν ως οδηγός ως προς την πορεία της αναζήτησης. Η επιλογή συγκεκριμένου τύπου εξαρτάται από το αντίστοιχο μοντέλο θεμάτων.

7.3 Παραγοντοποίηση Ιδιαζουσών Τιμών (SVD)

Η Παραγοντοποίηση Ιδιαζουσών Τιμών (Singular Value Decomposition ή SVD) είναι μία ισχυρή τεχνική για μείωση διαστάσεων ενός χώρου. Το κύριο θέμα σε μία SVD είναι να βρεθεί ένας χαμηλότερης διάστασης χώρος χαρακτηριστικών, με τον οποίο θα αναπαρίστανται οι «έννοιες» (concepts), και η επιρροή κάθε έννοιας ως προς τη δεδομένη συλλογή να είναι υπολογίσιμη. Επειδή η SVD επιτρέπει την αυτοματοποιημένη παραγωγή σημασιολογικών «εννοιών» σε χώρο χαμηλότερης διάστασης, μπορεί να χρησιμοποιηθεί ως βάση για λανθάνουσα-σημασιολογική ανάλυση (latent-semantic analysis) [12].

Το σκεπτικό της τεχνικής βασίζεται στη δυνατότητα αποδόμησης ενός πίνακα σε τρεις άλλους ως εξής:

$$A = U\lambda V^T$$

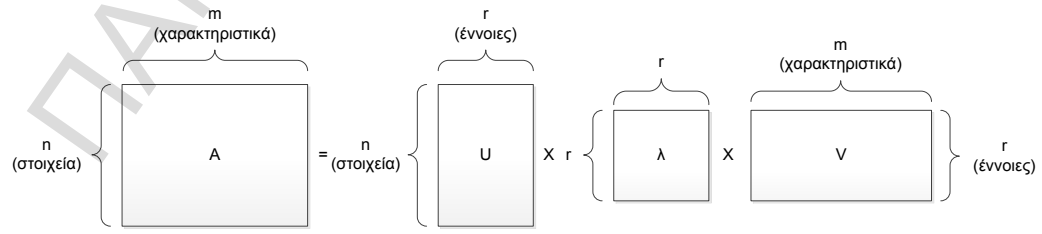
όπου A είναι ένας πίνακας με n στοιχεία και m χαρακτηριστικά,

U είναι ένας πίνακας με n στοιχεία και r έννοιες,

λ είναι ένας διαγώνιος πίνακας μεγέθους $r \times r$, όπου κάθε στοιχείο της διαγωνίου δείχνει την επιρροή της αντίστοιχης έννοιας, και

V είναι ένας πίνακας με m χαρακτηριστικά και r έννοιες.

Η παρακάτω εικόνα παρουσιάζει την τεχνική. Ο διαγώνιος πίνακας λ περιέχει θετικές τιμές σε φθίνουσα διάταξη. Ο πίνακας U μπορεί να θεωρηθεί ως ένας πίνακας ομοιότητας (similarity matrix) στοιχείων-εννοιών, ενώ ο πίνακας V ως ένας πίνακας ομοιότητας χαρακτηριστικών-εννοιών.



Εικόνα 6: Τεχνική SVD. Ένας πίνακας στοιχείων επί χαρακτηριστικών μπορεί να διασπαστεί σε τρεις: ένας με στοιχεία επί έννοιες, ένας διαγώνιος που υποδεικνύει την επιρροή των εννοιών και ένας με έννοιες επί χαρακτηριστικά.

Για να μπορέσει να υπολογιστεί η SVD ενός ορθογώνιου πίνακα A, ως στήλες του U λαμβάνονται τα ιδιοδιανύσματα του AA^T, ενώ ως στήλες του V τα ιδιοδιανύσματα του A^TA. Οι ιδιάζουσες τιμές (singular values) του λ είναι οι θετικές τετραγωνικές ρίζες των μη μηδενικών ιδιοτιμών τόσο του AA^T όσο και του A^TA.

Οι r ιδιοτιμές του λ είναι σε φθίνουσα διάταξη. Επομένως, ο αρχικός πίνακας A μπορεί να προσεγγιστεί διατηρώντας μόνο $k < r$ από τις ιδιοτιμές. Η περικομμένη SVD δημιουργεί μία βαθμού k προσέγγιση του A, τέτοια ώστε ο $A_k = U_k\lambda_k V_k^T$. Ο A_k είναι ο κοντινότερος βαθμού k πίνακας σε σχέση με τον A. Ο όρος «κοντινότερος» σημαίνει ότι ο A_k ελαχιστοποιεί το άθροισμα των τετραγώνων των διαφορών των στοιχείων A και A_k. Ο περικομμένος SVD είναι μία αναπαράσταση της λανθάνουσας δομής σε έναν μειωμένο χώρο διάστασης k, που γενικά σημαίνει μείωση του θορύβου στα χαρακτηριστικά.

Η τεχνική SVD χρησιμοποιείται καιρό στα συστήματα συστάσεων. Δύο από τους τρόπους που έχουν περιγραφεί για τη χρήση της είναι, πρώτον, η αποκάλυψη των λανθανουσών συσχετίσεων μεταξύ των χρηστών και των στοιχείων, όπου πρώτα συμπληρώνονται τα μηδενικά του πίνακα χρηστών-στοιχείων με τη μέση βαθμολογία μ και έπειτα κανονικοποιούνται αφαιρώντας από κάθε βαθμολογία τη μέση τιμή του εκάστοτε χρήστη b_u.

Ένα σημαντικό πλεονέκτημα της τεχνικής SVD είναι ότι έχουν υπάρξει αλγόριθμοι που δεν απαιτούν τον επανυπολογισμό του μοντέλου σε περίπτωση αλλαγών στους πίνακες, αλλά λειτουργούν προσθετικά βασιζόμενοι στα προηγούμενα δεδομένα. Να σημειωθεί πάντως ότι η τεχνική έχει κατηγορηθεί για υπερπροσαρμογή, καθώς συχνά τα δεδομένα των πινάκων είναι αραιά απαιτώντας την συμπλήρωση των μηδενικών τιμών με άλλες που μπορεί να μην είναι αληθείς. Το ίδιο ισχύει γενικότερα με τα μοντέλα παραγοντοποίησης πινάκων (matrix factorization models), που πολλές φορές χρησιμοποιούνται έναντι της SVD.

Τα μοντέλα παραγοντοποίησης πινάκων αντιστοιχίζουν τόσο τους χρήστες όσο και τα στοιχεία σε έναν κοινό χώρο, f διαστάσεων, λανθανόντων παραγόντων. Ο χώρος είναι τέτοιος ώστε οι αλληλεπιδράσεις των χρηστών με τα στοιχεία να μοντελοποιούνται ως εσωτερικά γινόμενα. Ο λανθάνων χώρος προσπαθεί να εξηγήσει τις βαθμολογίες χαρακτηρίζοντας τόσο τα στοιχεία όσο και τους χρήστες ως παράγοντες, που προκύπτουν από τις αναδράσεις των χρηστών. Για παράδειγμα, όταν τα στοιχεία είναι ταινίες, οι παράγοντες ενδέχεται να μετρούν προφανείς διαστάσεις, όπως κωμωδία έναντι δράματος, βαθμός δράσης ή προσανατολισμός για παιδιά, αλλά και λιγότερο προφανείς διαστάσεις, όπως η εμβάθυνση στην ανάπτυξη του χαρακτήρα, ή ακόμα και άλλες ανερμήνευτες διαστάσεις.

Επομένως, κάθε στοιχείο i σχετίζεται με ένα διάνυσμα $Q_i \in \mathbb{R}^r$, γραμμή του πίνακα $Q = \sqrt{\lambda_k} \cdot V_k{}'$ και κάθε χρήστης u σχετίζεται με ένα διάνυσμα $P_u \in \mathbb{R}^r$, γραμμή του πίνακα $P = U_k \cdot \sqrt{\lambda_k}$. Για ένα δεδομένο στοιχείο i τα μέλη του Q_i μετρούν την έκταση με την οποία το στοιχείο καταλαμβάνει εκείνους τους παράγοντες, είτε θετικά είτε αρνητικά. Για έναν δεδομένο χρήστη u τα μέλη του P_u μετρούν την έκταση του ενδιαφέροντος του χρήστη ως προς τα στοιχεία που έχουν υψηλά τους αντίστοιχους παράγοντες. Το εσωτερικό γινόμενο που προκύπτει, $q_i^T p_u$, αποτυπώνει την αλληλεπίδραση μεταξύ του χρήστη u και του στοιχείου i, δηλαδή το συνολικό ενδιαφέρον του χρήστη στα χαρακτηριστικά του στοιχείου. Η τελική βαθμολογία δημιουργείται από την επιπλέον πρόσθεση της μέσης τιμής των βαθμολογιών του χρήστη. Άρα, η βαθμολογία προβλέπεται από τον εξής κανόνα:

$$\widehat{r_{ui}} = b_u + Q_i^T P_u$$

7.4 Παραγωγή εγγράφων

Αξιοποιώντας τη σχεσιακή βάση δεδομένων, όπου εισήχθησαν τα δεδομένα του συστήματος συστάσεων, τα χαρακτηριστικά μπορούν να εμφανίζονται σε ξεχωριστούς πίνακες ώστε στη συνέχεια να σχετιστούν με τον πίνακα των στοιχείων. Αντίστοιχα, τα παραγόμενα έγγραφα θα αποθηκεύονται και αυτά σε ξεχωριστό πίνακα ο οποίος επίσης θα συνδέεται με τον πίνακα των στοιχείων.

Κάθε τύπος εγγράφου προκύπτει από τα χαρακτηριστικά από τα οποία αποτελείται. Η επιλογή των χαρακτηριστικών που θα συντελούν τον κάθε τύπο εγγράφου δεν πρέπει να γίνεται ανεξέλεγκτα, αλλά βάσει σχεδιασμού, κατά την οποία ο χρήστης επιλέγει από ποια γνωρίσματα

εξάγονται οι τιμές για τους όρους των εγγράφων. Επειδή η συλλογή των πληροφοριών γίνεται σε βάση δεδομένων, προτείνεται να δημιουργηθεί ένας πίνακας στον οποίο κάθε πλειάδα να αφορά ένα στοιχείο και να χρησιμοποιούνται οι προσχεδιασμένοι τύποι έγγραφων ως γνωρίσματα του πίνακα αυτού.

Κάθε έγγραφο ουσιαστικά είναι μία λίστα από χαρακτηριστικά ενός στοιχείου. Οι όροι δεν είναι απαραίτητο να είναι τύπου κειμένου σε πρώτη φάση. Απεναντίας, προτείνεται να χρησιμοποιούνται οι κωδικοί των πραγματικών όρων, όπου αυτό είναι δυνατόν, ώστε να αποφεύγεται η παρερμήνευσή τους. Σε τέτοιες περιπτώσεις, και γενικότερα όποτε υπάρχει πιθανότητα εμφάνισης ίδιας τιμής από διαφορετικά γνωρίσματα, χρησιμοποιείται ως πρόθεμα το όνομα του πίνακα, σχηματίζοντας με αυτόν τον τρόπο τον τελικό όρο. Προφανώς κάτι τέτοιο δεν είναι δυνατό όταν το συσχετιζόμενο πεδίο δε διαθέτει ξεχωριστό κωδικό. Στην περίπτωση αυτή, εφόσον το πεδίο είναι ήδη κείμενο, μπορεί να χρησιμοποιηθεί αυτούσια η τιμή του ως έγγραφο.

7.4.1 Προετοιμασία πινάκων

Κάποιοι από τους πίνακες που περιγράφουν χαρακτηριστικά των ταινιών περιέχουν πλεονασμούς ή είναι σε ακατάλληλη μορφή για να αποτελέσουν όρους εγγράφων. Γι' αυτό πραγματοποιούνται οι παρακάτω μετατροπές:

- Αποκοπή πρώτης λέξης στον πίνακα colorinfo:

```
INSERT colorinfo (movieid,color) (
      SELECT DISTINCT movieid,substring_index(color,' ',1)
      FROM oldcolorinfo
);
```

- Αποκοπή χρονολογίας στον πίνακα releasedates:

```
INSERT releasedates (movieid,releasedate) (
      SELECT DISTINCT(movieid) movieid,substring_index(releasedate,'-
',1) FROM oldreleasedates
);
```

- Δημιουργία ενδιάμεσων πινάκων:

```
INSERT runningtimes (runningtime) (
      SELECT DISTINCT time
      FROM oldrunningtimes
);

INSERT movies2runningtimes (movieid,runningtimeid)(
      SELECT movieid,runningtimeid
      FROM runningtimes, oldrunningtimes
      WHERE runningtime=time
);
```

- Εντολή αποκοπής κωδικού του πίνακα mpaaratings:

```
INSERT mpaaratings (movieid,reasontext)(
      SELECT movieid, SUBSTRING(SUBSTRING_INDEX(reasontext,' ',2),6)
      FROM oldmpaaratings
      WHERE reasontext LIKE 'Rated%'
);
```

- Εντολή συσχέτισης βαθμολογιών imdb στις κοινές ταινίες:

```
CREATE TABLE ratings
SELECT ml_id,cast(rank AS DECIMAL(2,1)) AS ratings
FROM oldratings,ml2imdb
WHERE movieid=imdb_id;
```

7.4.2 Υλοποίηση

Σκοπός είναι η εξαγωγή θεματικών χαρακτηριστικών σταθερού μεγέθους για κάθε στοιχείο. Για να επιτευχθεί αυτό πρέπει πρώτα να παραχθούν έγγραφα και αυτόν ακριβώς το σκοπό επιτελεί το παρακάτω πρόγραμμα. Η διαδικασία αυτή αυτοματοποιήθηκε ώστε να μπορεί να λειτουργήσει και για άλλες βάσεις δεδομένων, γι' αυτό σχεδιάστηκε και υλοποιήθηκε πρόγραμμα JAVA με 2 κλάσεις:

1. DocumentMaker.java: Η συγκεκριμένη κλάση αποσκοπεί στην ανάγνωση των παραμέτρων ώστε να κατασκευάσει τους τύπους εγγράφων χρησιμοποιώντας τη δεύτερη κλάση.
2. DocumentType.java: Η κλάση αυτή αναπαριστά τους τύπους των εγγράφων, τα οποία αποθηκεύονται σε μία βάση δεδομένων. Κάθε τύπος δεδομένου σχετίζεται με τα απαραίτητα χαρακτηριστικά των στοιχείων ενός συστήματος συστάσεων, τα οποία εντοπίζονται, μέσω των παραμέτρων του προγράμματος, επίσης σε βάση δεδομένων και πρέπει να σχετίζονται με τον πίνακα στοιχείων.

7.4.3 Κώδικας

- DocumentMaker.java

```
package efraimidis.topic;

import java.sql.SQLException;

/**
  * Main class, which receives the parameters of the program, like the
  * name of the document type, the number of the relations and, for
  * each relation, the names needed to find the related attributes. In
  * addition to that, a boolean value is needed to define the need of
  * a prefix if the values are numeric etc.
  */
public class DocumentMaker {
      public static void main(String[] args) throws SQLException {

            int i = 0;
            DocumentType.createConnection(args[i++]);
            do {
                  String documentName = args[i++];
                  int numberOfAssociations = Integer.parseInt(
                        args[i++]
                  );
                  if(numberOfAssociations>0) {
                        DocumentType documentType = new DocumentType(
                              documentName,
                              args[i],
                              args[i+1],
                              args[i+2],
                              args[i+3],
                              args[i+4],
                              args[i+5],
                              Boolean.parseBoolean(args[i+6])
                        );
                        i+=7;
                        for(int j=1;j<numberOfAssociations;j++) {
                              documentType.addAssociation(
                                    args[i],
```

```
                                        args[i+1],
                                        args[i+2],
                                        args[i+3],
                                        args[i+4],
                                        args[i+5],
                                        Boolean.parseBoolean(args[i+6])
                                );
                                i+=7;
                          }
                          documentType.insertDocuments();
                    } else {
                          System.out.println("Not enough associations");
                    }
              } while(i<args.length);
        }
}
```

- DocumentType.java

```
package efraimidis.topic;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map.Entry;

/**
 * Represents a type of topic modeling documents in recommender
 * systems, i.e. an attribute of a database table where each tuple
 * describes an item of recommendation. Like every database table's
 * attribute, a {@code DocumentType} has a name and it can be assigned
 * multiple values. Each value is mapped to an item, using a {@link
 * HashMap} of documents, where each document is a {@link String}
 * constructed by terms separated by spaces. The terms of each
 * document are specified by {@link Association}s made between terms'
 * tables and the items' tables. Additionally a {@code DocumentType}
 * needs a database {@link Connection} from which it should collect
 * the required data.
 */
public class DocumentType {

      /**
       * Holds the common {@code Connection} among all {@code
       * DocumentType}s.
       * There is a special constructor, that contains the connection
       * string in its parameters, and must be used for the creation
```

```
	 * of the first {@code DocumentType} instance.
	 */
	static private Connection connection;

	/**
	 * Holds the {@code PreparedStatement} that puts the documents
	 * into their {@code DocumentType}'s column.
	 */
	final private PreparedStatement insertion;
	/**
	 * Holds the mappings between items and documents.
	 */
	private HashMap<String,String> documents = new
		HashMap<String,String>();

	/**
	 * Initializes the J/Connector driver when the class is firstly
	 * used.
	 */
	static {
		try {
			System.out.println("Initializing driver");
			Class.forName("com.mysql.jdbc.Driver");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

	/**
	 * Creates the common, among all the {@code DocumentType}s,
	 * database {@code Connection}.
	 */
	static public void createConnection(String connectionString)
		throws SQLException {
		if(connectionString!=null) {
			System.out.println("Creating database connection");
			connection =
				DriverManager.getConnection(connectionString);
		}
	}
```

```
	/**
	 * Constructs a {@code DocumentType} instance. The common {@code
	 * Connection} must be already instantiated in order to be used
	 *for the creation of the documents' table and for the addition
	 * of the corresponding column for this {@code DocumentType} in
	 * the database. When the table is ready, the {@link
	 * Association} array is iterated fetching the required
	 * information for the documents to be inserted in the table.
	 */
	public DocumentType(
		String name,
		String itemTable,
		String itemPK,
		String itemId,
		String termTable,
		String itemFK,
		String term,
		boolean prefix
	) throws SQLException {

		insertion = connection.prepareStatement(
			"insert documents set id=?,"+
			name+
			"=? on duplicate key update id=?,"+
			name+
			"=?"
		);
		DatabaseMetaData metadata = connection.getMetaData();
		if(!metadata.getTables(
				null,
				null,
				"documents",
				null
			).next()
		) {
			System.out.println("Creating table 'documents' with
column for the document type '"+name+"'");
			connection.createStatement().execute(
				"create table documents (id int key, "+
				name+
				" longtext)"
			);
		} else {
			System.out.println(
				"Table 'documents' already exists"
			);
```

```
                // Check if column exists.
                if(!metadata.getColumns(
                            null,
                            null,
                            "documents",
                            name
                      ).next()
                ) {
                      System.out.println("Adding column for the
document type '"+name+"'");
                      connection.createStatement().execute(
                            "alter table documents add "+
                            name+
                            " longtext"
                      );
                } else {

                      System.out.println("Fetching existing
documents of type "+name);
                      ResultSet tuples = connection
                            .createStatement().executeQuery(
                                  "select id,"+
                                  name+
                                  " from documents"
                      );
                      while(tuples.next()) {
                            String key = tuples.getString(1);
                            String value = documents.get(key);
                            if(value!=null) {
                                  documents.put(
                                        key,
                                        value+
                                        " "+
                                        tuples.getString(2)
                                  );
                            } else {
                                  documents.put(
                                        key,
                                        tuples.getString(2)
                                  );
                            }
                      }
                }
          }
```

```
            // Fetches terms from the given database association.
            addAssociation(
                  itemTable,
                  itemPK,
                  itemId,
                  termTable,
                  itemFK,
                  term,
                  prefix
            );
      }

      /**
       * Fetches the terms, described by the given association, which
       * populate documents. An association's value describes either a
       * whole document or just a term. In the first case no prefix
       * should be used, while in the second, i.e. when no value
       * contains space characters, a check is made to see whether
       * there are more than one associations for the current
       * {@code DocumentType}. If there are more, a prefix must be
       * used.
       */
      public void addAssociation(
            String itemTable,
            String itemPK,
            String itemId,
            String termTable,
            String itemFK,
            String term,
            boolean prefix
      ) throws SQLException {

            System.out.println(
                  "Adding terms from attribute "+
                  termTable+
                  "."+
                  term+
                  " for items table "+
                  itemTable+
                  ". Prefix is "+
                  prefix
            );
```

```
            ResultSet tuples = connection.createStatement()
                  .executeQuery(
                        "select "+itemTable+"."+itemId+","+
                        termTable+"."+term+
                        " from "+itemTable+","+termTable+
                        " where "+itemTable+"."+itemPK+"="+
                        termTable+"."+itemFK
            );
            while(tuples.next()) {
                  String key = tuples.getString(1);
                  String value = documents.get(key);
                  String newValue;
                  if(prefix) {
                        newValue = termTable+"_"+tuples.getString(2);
                  } else {
                        newValue = tuples.getString(2);
                  }
                  if(value!=null) {
                        documents.put(key, value+" "+newValue);
                  } else {
                        documents.put(key, newValue);
                  }
            }
      }

      /**
       * Inserts or updates the documents into the database column
       * that corresponds to the current {@code DocumentType} by using
       * the statement that was prepared at the constructor.
       */
      public void insertDocuments() throws SQLException {
            System.out.println("Inserting documents");
            for(Entry<String,String> entry:documents.entrySet()) {
                  insertion.setInt(1,
                        Integer.parseInt(entry.getKey()));
                  insertion.setInt(3,
                        Integer.parseInt(entry.getKey()));
                  insertion.setString(2, entry.getValue());
                  insertion.setString(4, entry.getValue());
                  insertion.executeUpdate();
            }
      }
}
```

Να σημειωθεί ότι το πρόγραμμα χρησιμοποιεί τη βιβλιοθήκη J/Connector της MySQL.

7.4.4 Εκτέλεση

Αφού τα δεδομένα βρίσκονται ήδη σε σχεσιακή βάση δεδομένων, επιλέχθηκε να γίνει η συλλογή των εγγράφων σε έναν σχεσιακό πίνακα. Αναλυτικότερα για την εξαγωγή **κάθε είδους εγγράφου** πρέπει να δοθούν τα εξής στοιχεία:

1. το όνομα του είδους εγγράφου
2. το πλήθος των συσχετίσεων που παρέχουν όρους
3. για κάθε συσχέτιση:
    a. το όνομα του πίνακα των στοιχείων
    b. το όνομα του συσχετιζόμενου γνωρίσματος του πίνακα των στοιχείων
    c. το όνομα του προς εμφάνιση γνωρίσματος του πίνακα των στοιχείων
    d. το όνομα του πίνακα με τους όρους
    e. το όνομα του σχετιζόμενου γνωρίσματος του πίνακα με του όρους
    f. το όνομα του γνωρίσματος με τους όρους
    g. true για τη χρήση προθέματος, αλλιώς άλλη μη κενή τιμή.

Κάθε έγγραφο αποτελεί τιμή ενός γνωρίσματος του νέου πίνακα και χαρακτηρίζει ένα στοιχείο. Οι όροι κάθε εγγράφου παρέχονται από γνωρίσματα άλλων πινάκων που διαθέτουν επίσης ως γνώρισμα κλειδιά στοιχείων.

Χρησιμοποιώντας τη γραμματική ABNF μπορεί να περιγραφεί η δομή των παραμέτρων κατά την εκτέλεση της διαδικασίας με τυπικό τρόπο ως εξής:

```
παράμετροι = είδος *(SP είδος)
είδος = όνομα SP πλήθος 1*(SP στοιχείο SP όρος SP πρόθεμα)
όνομα = 1*VCHAR
πλήθος = 1*DIGIT
στοιχείο = πίνακας SP κλειδί SP χαρακτηριστικό
όρος = πίνακας SP κλειδί SP χαρακτηριστικό
πίνακας = 1*VCHAR
κλειδί = 1*VCHAR
χαρακτηριστικό = 1*VCHAR
πρόθεμα = "true" / "false"
```

Η εισαγωγή δεδομένων σταματάει όταν δε βρεθεί επόμενο όνομα είδους.

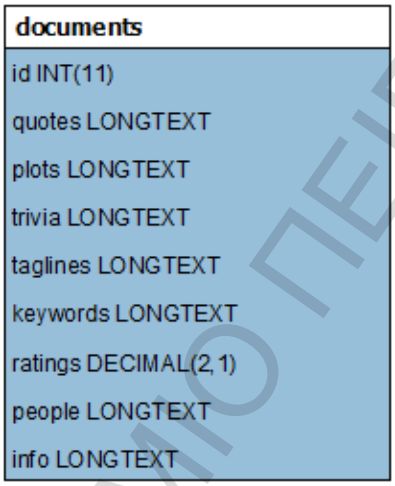
Στη συνέχεια πρέπει για κάθε στοιχείο του συστήματος συστάσεων να παραχθούν τα αντίστοιχα έγγραφα συλλέγοντας όρους από τα επιλεγμένα γνωρίσματα και να εισαχθούν σε έναν πίνακα με τόσες πλειάδες όσα είναι τα στοιχεία και γνωρίσματα όσα είναι τα είδη των εγγράφων.

Για το παράδειγμα με την ταινιοθήκη χρησιμοποιείται η τομή των βάσεων IMDb και MovieLens. Συγκεκριμένα επιλέγονται για έγγραφα χρησιμοποιώντας τις εξής παραμέτρους:

- trivia 1 ml2imdb imdb_id ml_id trivia movieid triviatext false
- quotes 1 ml2imdb imdb_id ml_id quotes movieid quotetext false
- plots 1 ml2imdb imdb_id ml_id plots movieid plottext false
- taglines 1 ml2imdb imdb_id ml_id taglines movieid taglinetext false
- keywords 1 ml2imdb imdb_id ml_id keywords movieid keyword false
- people 9 ml2imdb imdb_id ml_id movies2writers movieid writerid true ml2imdb imdb_id ml_id movies2proddes movieid proddesid true ml2imdb imdb_id ml_id movies2directors movieid directorid true ml2imdb imdb_id ml_id movies2costdes movieid costdesid true ml2imdb imdb_id ml_id movies2producers movieid producerid true ml2imdb imdb_id ml_id movies2actors movieid actorid true ml2imdb imdb_id ml_id movies2cinematgrs movieid cinematid true ml2imdb imdb_id ml_id movies2editors movieid editorid true ml2imdb imdb_id ml_id movies2composers movieid composerid true

- info 8 ml2imdb imdb_id ml_id genres movieid genre false ml2imdb ml_id ml_id ml_moviegenres movieid genre false ml2imdb imdb_id ml_id movies2prodcompanies movieid prodcompanyid true ml2imdb imdb_id ml_id movies2countries movieid countryid true ml2imdb imdb_id ml_id colorinfo movieid color false ml2imdb imdb_id ml_id releasedates movieid releasedate true ml2imdb imdb_id ml_id movies2runningtimes movieid runningtimeid true ml2imdb imdb_id ml_id mpaaratings movieid reasontext false
- ratings 1 ml2imdb ml_id ml_id newratings ml_id rating false (και μετατροπή του πεδίου σε decimal(2,1))

Από τις παραπάνω εκτελέσεις του κώδικα προκύπτει ο παρακάτω πίνακας στη βάση δεδομένων, όπου το γνώρισμα id σχετίζεται με μία ταινία, δηλαδή με τις τιμές του πίνακα στοιχείων του συστήματος συστάσεων:



Εικόνα 7: Ο πίνακας documents με τα παραχθέντα έγγραφα.

Με αυτόν τον τρόπο ουσιαστικά περιγράφεται κάθε ταινία από 8 έγγραφα, τα οποία μπορούν στη συνέχεια να αξιοποιήσουν τεχνικές μοντελοποίησης θεμάτων ώστε να βρεθούν παρόμοιες ταινίες και να προταθούν στους κατάλληλους χρήστες.

8 Συμπεράσματα

Τα συστήματα συστάσεων αποτελούν ανοιχτό χώρο έρευνας και, ως τέτοια, χαρακτηρίζονται από ποικιλομορφία και μεταβλητότητα. Η πληθώρα υλοποιήσεων που εντοπίστηκε στη βιβλιογραφία, κάθε άλλο παρά ευκαταφρόνητη είναι. Παρόλα αυτά, τα προβλήματα που παρουσιάζουν, είτε στο σύνολό τους είτε κάθε είδος ξεχωριστά, δημιουργούν την ανάγκη για προτάσεις νέων λύσεων.

Ένα από τα σημαντικότερα προβλήματα είναι τα αραιά δεδομένα, καθώς η έλλειψη βαθμολογιών, στις οποίες τα περισσότερα συστήματα συστάσεων βασίζουν τις προτάσεις τους, είναι το ίδιο το πρόβλημα που προσπαθούν να λύσουν. Στην παρούσα εργασία υλοποιήθηκε ένα τέτοιο σύστημα με σκοπό τη διαπίστωση αυτού του φαινομένου.

Για την επίτευξη αυτού του σκοπού έγινε σύζευξη των δύο γνωστότερων βάσεων δεδομένων με θέμα τις κινηματογραφικές ταινίες, δηλαδή της IMDB και της MovieLens. Με αυτόν τον τρόπο δημιουργήθηκε μία πλούσια βάση δεδομένων, που συνδυάζει τα χαρακτηριστικά των κοινών ταινιών που διαθέτει η IMDB με τις βαθμολογίες των χρηστών και τα δημογραφικά τους δεδομένα, όπως παρέχονται ανωνυμοποιημένα από τη MovieLens.

Από την πειραματική υλοποίηση που πραγματοποιήθηκε στα δεδομένα που συλλέχτηκαν, τα οποία ήταν αραιά αφού μόλις το 5% του πίνακα βαθμολογιών περιείχε δεδομένα, διαπιστώθηκε πως το σύστημα συστάσεων ανταποκρίθηκε ικανοποιητικά. Συγκεκριμένα, οι εκτιμήσεις του συστήματος κατά μέσο όρο απέκλινε πάντα λιγότερο του 1 αστεριού (με μέγιστη βαθμολογία τα 5 αστέρια). Παρόλα αυτά και στις 10 κατατμήσεις του συνόλου δεδομένων παρατηρήθηκε ότι υπήρχαν αποκλίσεις 4 και 5 αστεριών, δηλαδή δεν μπορούσε να εγγυηθεί καλή προσέγγιση σε όλο το σύνολο των βαθμολογιών.

Στη συνέχεια, συντάχθηκε σε θεωρητικό επίπεδο μία λύση που να μη βασίζεται στις αξιολογήσεις, αλλά στην εξαγωγή θεμάτων από τα χαρακτηριστικά των προς πρόταση στοιχείων. Γι' αυτόν το σκοπό υλοποιήθηκε πρόγραμμα που να παράγει έγγραφα αξιοποιώντας τα χαρακτηριστικά των ταινιών, ώστε στη συνέχεια χρησιμοποιώντας μεθοδολογίες μοντελοποίησης θεμάτων να κατασκευαστεί ένας συμπαγής πίνακας θεμάτων που να περιγράφουν αποδοτικότερα τα στοιχεία.

9 Βιβλιογραφία

[1] F. Ricci, L. Rokach and B. Sapira, "Introduction to Recommender Systems Handbook," in *Recommender Systems Handbook*, New York, Springer, 2011, pp. 1-35.

[2] C. Desrosiers and G. Karypis, "A Comprehensive Survey of Neighborhood-based Recommendation Methods," in *Recommender Systems Handbook*, New York, Springer, 2011, pp. 107-144.

[3] J. A. Konstan and J. Riedl, "Recommender systems: from algorithms to user experience," *User Modeling and User-Adapted Interaction,* no. 22, pp. 101-123, 2012.

[4] J. Bobadilla, F. Ortega, A. Hernando and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems,* pp. 109-132, 2013.

[5] X. Su and T. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence,* 2009.

[6] G. Adomavicius and A. Tuzhilin, "Context-Aware Recommender Systems," in *Recommender Systems Handbook*, New York, Springer, 2011, pp. 217-253.

[7] P. Lops, M. de Gemmis and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," in *Recommender Systems Handbook*, New York, Springer, 2011, pp. 70-102.

[8] O. Arazy, N. Kumar and B. Shapira, "Improving Social Recommender Systems," *IT Professional,* vol. 11, no. 4, pp. 38-44, 2009.

[9] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Handbook of latent semantic analysis*, 2007, pp. 424-440.

[10] D. M. Blei, Y. A. Ng and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research 3,* pp. 993-1022, 2003.

[11] D. M. Blei, "Probabilistic topic models," *Communications of the ACM,* vol. 55, no. 4, pp. 77-84, 2012.

[12] X. Amatriain, A. Jaimes, N. Oliver and J. M. Pujol, "Data Mining Methods for Recommender Systems," in *Recommender Systems Handbook*, New York, Springer, 2011, pp. 39-71.

[13] GroupLens, "MovieLens 1M," 2011 8 2011. [Online]. Available: http://www.grouplens.org/system/files/ml-1m.zip. [Accessed 29 12 2012].

[14] D. H. Crocker and P. Overell, "RC5234 - Augmented BNF for Syntax Specifications: ABNF," January 2008. [Online]. Available: https://tools.ietf.org/html/rfc5234. [Accessed 1 August 2013].

[15] V. Peralta, "Matching of MovieLens and IMDb Movie Titles," Technical Report, Laboratoire PRiSM, Université de Versailles, Versailles, France, 2007.

[16] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Application of dimensionality reduction in recommender systems - a case study," in *ACM WebKDD Workshop*, 2000.

ΠΑΡΑΡΤΗΜΑ Α

1 Γραμματική ABNF

Για την καταγραφή της δομής ενός οποιουδήποτε εγγράφου μπορεί να χρησιμοποιείται μία παραλλαγή της γραμματικής BNF (Backus Naur Form), η Augmented BNF (ABNF) [14]. Σύμφωνα με την προδιαγραφή της είναι δυνατόν να περιγράφονται τα έγγραφα με τυπικό τρόπο ώστε να καθορίζεται η σύνταξή τους λιτά και δίχως παρερμηνείες.

Ως γνωστόν τα έγγραφα εν γένει αποτελούνται από χαρακτήρες. Κάθε χαρακτήρας στην ABNF ονομάζεται **τερματική τιμή (terminal value)** και θεωρείται ως μία αντιστοίχιση με έναν μη αρνητικό αριθμό, όπως συμβαίνει για παράδειγμα στον κώδικα ASCII. Για λόγους ευκολίας αντί των αριθμητικών τιμών μπορούν, όποτε είναι τεχνικά εφικτό, να χρησιμοποιηθούν απευθείας οι χαρακτήρες εισάγοντάς τους μόνους τους ή κατά ομάδες μέσα σε εισαγωγικά (“ ”). Επίσης για διάφορες τιμές χαρακτήρων που χρησιμοποιούνται συχνά έχουν προδιαγραφτεί συγκεκριμένες συντομογραφίες, που μπορούν να χρησιμοποιηθούν εναλλακτικά.

Στη βάση της η γραμματική ABNF αποτελείται από **κανόνες (rules)**, τα ονόματα των οποίων είναι σειρές χαρακτήρων που ξεκινούν με γράμμα του αλφάβητου και ακολουθούνται από συνδυασμούς γραμμάτων του αλφάβητου, αριθμητικών ψηφίων και κάτω παυλών. Επιπλέον στα ονόματα δεν διακρίνονται τα κεφαλαία από τα πεζά γράμματα, ενώ όταν ένας κανόνας εμφανίζεται σε φυσικό κείμενο εισάγεται μέσα σε γωνιώδεις αγκύλες (< >).

Το όνομα κάθε κανόνα αντιστοιχεί σε μία **δομή (form),** δηλαδή μία σειρά από συσχετισμένα **στοιχεία (elements)**, που χρησιμοποιώντας τους παρεχόμενους **τελεστές (operators)** παράγουν ένα σύνολο από επιτρεπόμενους συνδυασμούς τερματικών τιμών. Κάθε στοιχείο της ABNF είτε υποδεικνύει μία τερματική τιμή είτε περιέχει έναν από τους κανόνες, που με τη σειρά του θα υποδείξει καμία, μία ή συνδυασμούς τερματικών τιμών που επιτρέπεται να χρησιμοποιηθούν στον κανόνα αντί του στοιχείου.

Ο ορισμός ενός κανόνα γράφεται υπό μορφή ισότητας με το όνομα στα αριστερά του συμβόλου της ισότητας (=) και στα δεξιά του να ακολουθεί η δομή. Το τέλος του ορισμού του κανόνα υποδεικνύεται με την αλλαγή γραμμής, δηλαδή με τους χαρακτήρες carriage return και line feed. Όταν κάποιος κανόνας ξεπερνά τη μία γραμμή, οι επόμενες γραμμές του κανόνα τοποθετούνται σε εσοχή για λόγους εμφάνισης.

Οι τελεστές παρουσιάζονται στη δομή ενός κανόνα ώστε να μετασχηματίσουν τα στοιχεία κάνοντας δυνατή την περιγραφή συνθετότερων συνδυασμών με πιο συμπυκνωμένους κανόνες. Οι τελεστές που διατίθενται στη γραμματική ABNF είναι οι εξής:

1. ένωσης (concatenation): δομή1 δομή2

   Με έναν οποιοδήποτε κενό χαρακτήρα (space, tab κ.α.) μεταξύ δύο δομών ενώνονται τα σύνολα των επιτρεπόμενων συνδυασμών τερματικών τιμών σχηματίζοντας ένα ευρύτερο.

2. εναλλαγής (alternatives): στοιχείο1 / στοιχείο2

   Ο τελεστής εναλλαγής (/) μεταξύ δύο στοιχείων επιτρέπει τη χρήση ενός εκ των δύο στοιχείων.

3. αύξουσας εναλλαγής (incremental alternatives): κανόνας =/ δομή

   Ο τελεστής αύξουσας εναλλαγής μεταξύ ενός προϋπάρχοντα κανόνα και μίας δομής προσθέτει στον κανόνα τη δομή ως εναλλακτική.

4. εύρους (value range alternatives): τερματική_τιμή1 - τερματική_τιμή2

   Ο τελεστής εύρους (-) μεταξύ δύο τερματικών τιμών επιτρέπει τη χρήση ενός εκ των δύο τερματικών τιμών ή οποιασδήποτε ενδιάμεσής τους.

5. ομαδοποίησης (sequence group): (δομή)

   Ο τελεστής ομαδοποίησης μετατρέπει μία δομή σε στοιχείο.

6. μεταβλητής επανάληψης (variable repetition): <α>*<β>στοιχείο

   Ο τελεστής επανάληψης (<α>*<β>) πριν από ένα στοιχείο υποδηλώνει την εμφάνιση σε σειρά συνδυασμών τερματικών τιμών που παράγονται από το συγκεκριμένο στοιχείο τουλάχιστον όσες φορές υποδηλώνει η τιμή <α> και το πολύ όσες φορές

υποδηλώνει η <β>. Οι τιμές είναι προαιρετικές, με την έλλειψη της <α> να υποδηλώνει το μηδέν και την έλλειψη της <β> το άπειρο.

7. συγκεκριμένης επανάληψης (specific repetition): <α>στοιχείο

   Ο τελεστής συγκεκριμένης επανάληψης <α> πριν από ένα στοιχείο είναι ένας ακέραιος που δείχνει ακριβώς πόσες φορές απαιτείται να επαναληφθεί το στοιχείο.

8. προαιρετικής δομής (optional sequence): [δομή]

   Ο τελεστής προαιρετικής δομής κάνει δεκτή την έλλειψη συνδυασμού τερματικών τιμών.

9. σχόλιο (comment): ; σχόλιο

   Ό,τι ακολουθεί τον τελεστή του σχολίου δεν παράγει τερματικές τιμές.