Πανεπιστημιο Πατρων
Πολυτεχνικη Σχολη
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών
και Πληροφορικής

Διπλωματική Εργασία

Μελέτη και Υλοποίηση Αλγορίθμων Συνεργατικής Διήθησης, για την Ευφυή Παραγωγή Συστάσεων υπό Συνθήκες Κρύας Εκκίνησης

Χρυσάνθη Γ. Λαγοδήμου
Α.Μ. 4208

Επιβλέπων: Ιωάννης Γαροφαλάκης, Καθηγητής

Πάτρα, Νοέμβριος 2015

Περίληψη

Τα συστήματα παραγωγής συστάσεων είναι εργαλεία λογισμικού και τεχνικές που στοχεύουν στο να βοηθήσουν τους χρήστες να επιλέξουν ανάμεσα σε μια πληθώρα διαθέσιμων ειδών. Στην παρούσα διπλωματική εργασία έγινε μια επισκόπηση των διαφόρων προσεγγίσεων στον τομέα των συστημάτων παραγωγής συστάσεων. Εξετάστηκε κυρίως αυτή της συνεργατικής διήθησης καθώς κρίνεται μία από τις πιο επιτυχημένες. Ιδιαίτερη αναφορά έγινε στο πρόβλημα της κρύας εκκίνησης, αφού αποτελεί πρόκληση για την επιτυχή παραγωγή συστάσεων.

Υλοποιήθηκε ο αλγόριθμος Hierarchical Itemspace Rank (HIR), ο οποίος επιλέχθηκε λόγω της καλής συμπεριφοράς του στο πρόβλημα της κρύας εκκίνησης. Η επιτυχία του στον τομέα αυτό προκύπτει από την αξιοποίηση της εγγενούς ιεραρχικής δομής του χώρου των ειδών, μέσω της οποίας αντιμετωπίζει επιτυχώς τα προβλήματα της αραιότητας και της κρύας εκκίνησης και παράγει ποιοτικές συστάσεις.

Εκτός από τον ίδιο τον αλγόριθμο, υλοποιήθηκε και ένα demo, το οποίο τον χρησιμοποιεί για να προτείνει σε χρήστες ταινίες, χρησιμοποιώντας ένα ειδικό για το σκοπό αυτό σύνολο δεδομενων. Οι συστάσεις γίνονται βάσει των βαθμολογιών που έχουν ήδη δώσει οι χρήστες σε κάποιες ταινίες.

Η υλοποίηση έγινε σε Java στο λογισμικό LensKit. Το LensKit αποτελεί ένα ολοκληρωμένο σύστημα για την υλοποίηση, τη σύγκριση, την πειραματική αξιολόγηση αλλά και την έρευνα πάνω σε αλγορίθμους παραγωγής συστάσεων. Επιλέχθηκε λόγω των δυνατοτήτων που προσφέρει τόσο στον τομέα της υλοποίησης όσο και στο χειρισμό των απαιτούμενων δεδομένων.

Η παρούσα υλοποίηση έχει τη δυνατότητα να ενταχθεί σε μία νέα ενότητα υλοποιημένων αλγορίθμων στο LensKit, η οποία θα εξειδικεύεται στην αντιμετώπιση του προβλήματος της κρύας εκκίνησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συστήματα Παραγωγής Συστάσεων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : Συνεργατική Διήθηση, Κρύα Εκκίνηση, HIR, LensKit

Abstract

Recommender Systems are software tools and techniques that aim to help users choose between a wide range of items. This diploma thesis contains an overview of the various approaches in the field of recommender systems. Emphasis was given on the collaborative filtering approach, since it is considered to be one of the most successful ones, and on the Cold-Start problem, that is a challenge for the generation of successful recommendations.

Hierarchical Itemspace Rank (HIR) is the algorithm that was chosen to be implemented. HIR tackles successfully the cold-start problem. It exploits the innate hierarchical structure of the itemspace to face the sparsity and cold-start problems and to generate qualitative recommendations.

Except for the algorithm itself, a demo was implemented. This demo uses HIR to recommend movies to users, using a special for this purpose dataset. The recommendations are based on the users' ratings on some movies.

The algorithm was implemented in Java using LensKit software. LensKit is a complete system for the implementation, comparison, experimental evaluation and research on recommender algorithms. It was chosen due to its features that enable the implementation of algorithms and the handling of the required data.

This implementation can be part of a new module in LensKit that would be specialized in facing cold-start problem.

AREA: Recommender Systems
KEYWORDS: Collaborative Filtering, Cold-Start Problem, HIR, LensKit

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας ολοκληρώνεται ένας κύκλος σπουδών στο Τμήμα Μηχανικών Η/Υ και Πληροφορικής. Θα ήθελα να ευχαριστήσω όσους συνέβαλαν με τον τρόπο τους σε αυτό.

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ιωάννη Γαροφαλάκη για την εμπιστοσύνη που μου έδειξε με την ανάθεση της παρούσας διπλωματικής εργασίας, με την οποία μου έδωσε την ευκαιρία να ανακαλύψω και να ασχοληθώ με ένα τόσο ενδιαφέρον αντικείμενο.

Θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Αθανάσιο Νικολακόπουλο για την καθοδήγησή του στη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας καθώς και για τις γνώσεις που αποκόμισα κατά τη διάρκεια της. Όπως επίσης και για τις επεξηγήσεις του πάνω στη δημοσίευσή του και τον αλγόριθμο που προτείνεται σε αυτή, ο οποίος ήταν το αντικείμενο της υλοποίησης που έγινε στα πλαίσια αυτής της διπλωματικής εργασίας.

Ευχαριστώ τον Δημήτρη Λεβεντέα, ο οποίος με μύησε στην Python και τη Βασιλική Παρπαρούση, η οποία διάβασε το κείμενο της διπλωματικής μου και μου επεσήμανε ζητήματα στην έκφρασή του.

Ευχαριστώ τους φίλους μου Δημήτρη Λεβεντέα, Δώρα Πάνα, Βασιλική Παρπαρούση, Βασίλη Φράγκο, Τάσο Σιαφλέκη και Σοφοκλή Κυριακόπουλο που ήταν δίπλα μου όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Γιώργο και Σοφία, και τον αδερφό μου Γιάννη για την αγάπη και τη στήριξή τους στη διάρκεια της ζωής μου.

Χρυσάνθη Γ. Λαγοδήμου

Πάτρα, 10 Νοεμβρίου 2015

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή στους Αλγορίθμους Παραγωγής Συστάσεων

1.1 Συστήματα Παραγωγής Συστάσεων

Τα συστήματα παραγωγής συστάσεων είναι εργαλεία λογισμικού και τεχνικές που στοχεύουν στο να προτείνουν είδη που ενδιαφέρουν τους χρήστες.[25][1] Προέκυψαν από την ανάγκη διευκόλυνσης των χρηστών να επιλέξουν ανάμεσα από μια πληθώρα διαθέσιμων ειδών.

Με δεδομένα τα σύνολα χρηστών, ειδών και άμεσων ή έμμεσων βαθμολογιών των χρηστών για τα είδη, τα συστήματα παραγωγής συστάσεων προσπαθούν είτε να προβλέψουν τις βαθμολογίες των χρηστών για είδη με τα οποία δεν έχουν αλληλεπιδράσει είτε να προτείνουν σε κάποιο χρήστη μία λίστα ειδών που μπορεί να τον ενδιαφέρουν. [24]

Η επιλογή της μεθόδου βάσει της οποίας θα υλοποιηθεί ένας παραγωγός συστάσεων εξαρτάται από το σκοπό για τον οποίο θα χρησιμοποιηθεί και από τα διαθέσιμα δεδομένα. Τα συστήματα παραγωγής συστάσεων διακρίνονται σε έξι κατηγορίες [2]:

1. Βάσει περιεχομένου (Content-based): Τα συστήματα αυτά αναλύουν ένα σύνολο περιγραφών των ειδών, τα οποία έχουν βαθμολογηθεί από τον χρήστη και κατασκευάζουν ένα προφίλ των ενδιαφερόντων του βάσει των χαρακτηριστικών των ειδών. [22] Οι συστάσεις παράγονται μέσω της αντιστοίχισης των χαρακτηριστικών του χρήστη, όπως αυτά προέκυψαν από τα ενδιαφέροντά του, με αυτά των ειδών, βάσει των οποίων έχει υπολογιστεί και η ομοιότητά τους.

[1] Κύρια πηγή του παρόντος κεφαλαίου είναι το [25].

2. Συνεργατικής διήθησης (Collaborative filtering): Βασίζονται στην παρατήρηση ότι οι άνθρωποι συχνά επιλέγουν ανάμεσα στα διάφορα είδη βάσει των συστάσεων που παρέχονται από άλλους. Θεωρούν ότι αν ένας χρήστης στο παρελθόν είχε παρόμοιες προτιμήσεις με κάποιους άλλους χρήστες, τότε οι συστάσεις που προέρχονται από αυτούς θα τον ενδιαφέρουν και στο μέλλον. Είναι από τις πιο υλοποιημένες προσεγγίσεις [25] και κρίνεται από τις πιο επιτυχημένες.[24]

3. Δημογραφικά (Demographic): Προτείνουν με βάση τη χώρα που βρίσκεται ο χρήστης, το φύλο, την ηλικία του και άλλα στοιχεία που γνωρίζουν για αυτόν.

4. Βάσει γνώσης (Knowledge-based): Παρέχουν συστάσεις υπολογίζοντας κατά πόσο τα χαρακτηριστικά των ειδών καλύπτουν τις ανάγκες των χρηστών.

5. Βάσει κοινότητας (Community-based): Οι συστάσεις που παρέχουν σε κάποιο χρήστη βασίζονται στις προτιμήσεις των φίλων του καθώς αποδεικνύεται ότι οι άνθρωποι τείνουν να εμπιστεύονται περισσότερο τις συστάσεις που προέρχονται από φίλους τους από αυτές παρόμοιων αλλά άγνωστων σε αυτούς χρηστών. [27]

6. Υβριδικά (Hybrid): Αποτελούν μίξη των παραπάνω κατηγοριών, με τη μία κατηγορία να προσπαθεί να καλύψει τα μειονεκτήματα της άλλης. Για παράδειγμα, τα συστήματα συνεργατικής διήθησης δε μπορούν να προτείνουν είδη για τα οποία δεν έχουν καθόλου βαθμολογίες. Η μίξη ενός τέτοιου συστήματος με ένα που βασίζεται στο περιεχομένο μπορεί να παράξει συστάσεις και για τέτοια είδη.

Οι δύο πρώτες αποτελούν τις πιο σημαντικές κατηγορίες συστημάτων παραγωγής συστάσεων.

1.2 Παραγωγή Συστάσεων Βάσει Περιεχομένου

Η αρχιτεκτονική ενός τέτοιου συστήματος αποτελείται από τρία βασικά τμήματα:

- Αναλυτής Περιεχομένου (Content Analyzer) : Αυτό το τμήμα επεξεργάζεται τα είδη και τα μετασχηματίζει από την αρχική τους μορφή σε ένα χώρο χαρακτηριστικών, ο οποίος χρησιμοποιείται από τα επόμενα επίπεδα του συστήματος.

- Κατασκευαστής Προφίλ (Profile Learner) : Συλλέγει αντιπροσωπευτικά δεδομένα για τα ενδιαφέροντα του χρήστη με στόχο την κατασκευή ενός προφίλ του. Τα δεδομένα αυτά προκύπτουν από τα είδη που έχει βαθμολογήσει ο

χρήστης και γενικεύονται με τεχνικές μηχανικής μάθησης σε ένα μοντέλο που αντιπροσωπεύει τα ενδιαφέροντά του. [21]

- Τμημα Διήθησης (Filtering Component) : Το τμήμα αυτό αξιοποιεί το προφίλ του χρήστη για να προτείνει είδη που ταιριάζουν σε αυτό.

Οι τεχνικές αυτές έχουν κάποια πλεονεκτήματα σε σχέση με αυτές της συνεργατικής διήθησης:

- Είναι ανεξάρτητες από το σύνολο των χρηστών. Μόνο οι βαθμολογίες κάθε χρήστη είναι απαιραίτητες για την παραγωγή συστάσεων προς αυτόν. Αν το σύνολο χρηστών είναι μικρό, είναι πιο εύκολο να παραχθούν συστάσεις με αυτές τις τεχνικές παρά με αυτές της συνεργατικής δίηθησης. Οι τελευταίες για να παράξουν συστάσεις για κάποιον χρήστη απαιτούν την ύπαρξη χρηστών με παρόμοιες προτιμήσεις με αυτόν.

- Μπορούν ακόμα να παρέχουν και τους λόγους που οδήγησαν σε μία σύσταση με πιο άμεσο και κατανοητό τρόπο για τον χρήστη. Ο χρήστης μπορεί να έχει πρόσβαση στη λίστα των χαρακτηριστικών των ειδών, τα οποία ήταν αιτία για τη σύσταση που έλαβε. Αυτό του δίνει τη δυνατότητα να επιλέξει πιο εύκολα αν θα την εμπιστευτεί.

- Αντιμετωπίζουν πιο δίκαια τα νέα είδη. Τα νέα είδη, όπως και όλα τα υπόλοιπα, προτείνονται βάσει του περιεχομένου τους και δε χρειάζεται να έχουν βαθμολογηθεί από κάποιο χρήστη.

Ταυτόχρονα, όμως, έχουν και κάποιους περιορισμούς. Ανεξάρτητα από τον τρόπο εξαγωγής των χαρακτηριστικών των ειδών, κάθε είδος μπορεί να συσχετισθεί με ένα πεπερασμένο πλήθος χαρακτηριστικών. Επιπλέον, αν δεν υπάρχει αρκετή πληροφορία για το περιεχόμενο και τα χαρακτηριστικά τους, δεν είναι δυνατή η παραγωγή επιτυχών συστάσεων. Επίσης, δεν έχουν τη δυνατότητα να προτείνουν κάτι πέρα από αυτά που θα μπορούσε να ανακαλύψει μόνος του ο χρήστης. Ένας αξιολογητής ενός αλγορίθμου αυτής της κατηγορίας θα θεωρούσε κακή μία σύσταση, η οποία δε θα ταίριαζε με το προφίλ του χρήστη. Είναι πιθανό όμως μια τέτοια σύσταση να ενδιέφερε τελικά τον χρήστη. Τέλος, αν δεν υπάρχουν αρκετές βαθμολογίες στο σύστημα, δε μπορούν να εξάγουν τις προτιμήσεις των χρηστών και να προτείνουν αξιόπιστες συστάσεις, ειδικά σε νέους χρήστες με λίγες βαθμολογίες.

1.3 Η Συνεργατική Διήθηση

Τα συστήματα συνεργατικής δίηθησης δεν αντιμετωπίζουν κάποια από τα μειονεκτήματα των συστημάτων που βασίζονται στο περιεχομένο επειδή βασίζονται στις βαθμολογίες των χρηστών. Μπορούν να προτείνουν είδη των οποίων το περιεχομένο δεν είναι γνωστό ή δε μπορεί να ανακτηθεί, αλλά και είδη που ανήκουν σε

πολύ διαφορετικές κατηγορίες, αν ο χρήστης έχει δείξει ενδιαφέρον για αυτές. Επιπλέον, βασίζονται στην ποιότητα των ειδών όπως αυτή έχει διαμορφωθεί από τις βαθμολογίες των χρηστών και όχι στο περιεχομένο, το οποίο πολλές φορές δεν αποτελεί κριτήριο για την ποιότητα ενός είδους. [25]

Τα συστήματα αυτά διαχωρίζονται σε δύο γενικές κατηγορίες: σε αυτά που παράγουν συστάσεις βάσει γειτονιάς (neighborhood-based) και εκείνα που το πράττουν βάσει μοντέλου (model-based).

1.3.1 Συνεργατική Διήθηση Βάσει Γειτονιάς

Η συνεργατική διήθηση βάσει γειτονιάς είναι αρκετά δημοφιλής λόγω της απλότητας, της αποτελεσματικότητας και της ικανότητάς της να παράγει ακριβείς και προσωποποιημένες συστάσεις.[25] Τα συστήματα που ακολουθούν αυτή την προσέγγιση χρησιμοποιούν απευθείας τις βαθμολογίες των χρηστών για να προβλέψουν τις βαθμολογίες που θα έδιναν οι χρήστες για τα νέα είδη. Αυτές οι προβλέψεις αποτελούν τον οδηγό για την παραγωγή συστάσεων. Η χρήση των βαθμολογιών μπορεί να γίνει είτε με τρόπο που βασίζεται στους χρήστες (user-based) είτε στα είδη (item-based). Τα πρώτα εκτιμούν το ενδιαφέρον του χρήστη για ένα είδος χρησιμοποιώντας τις βαθμολογίες άλλων χρηστών (γείτονες) με τους οποίους έχει βαθμολογήσει με παρόμοιο τρόπο τα κοινά τους είδη. Στα βασιζόμενα στα είδη, η πρόβλεψη για τη βαθμολογία ενός χρήστη γίνεται με βάση το πώς έχει βαθμολογήσει παρόμοια με το υπό εξέταση είδος. Δύο είδη θεωρούνται παρόμοια αν πολλοί χρήστες τα έχουν βαθμολογήσει με παρόμοιο τρόπο.

Τα συστήματα αυτά υστερούν όσον αφορά την πρόβλεψη βαθμολογιών σε σχέση με τους καλύτερους αλγορίθμους που βασίζονται στο μοντέλο.[17, 28] Όμως έχει γίνει πλέον σαφές ότι η ακρίβεια των προβλέψεων δεν είναι ο μόνος παράγοντας που διασφαλίζει την αποτελεσματικότητα του συστήματος. Επιτυχημένη πρόβλεψη δεν είναι μόνο αυτή που προτείνει στο χρήστη απλά ένα νέο είδος, αλλά και αυτή (και αυτό είναι το πιο δύσκολο) που του δίνει την ευκαιρία να ανακαλύψει είδη ή κατηγορίες που μόνος του δε θα το έκανε.[11] Είναι πιο πιθανό να προτείνουν σε κάποιο χρήστη κάποιο είδος που δεν έχει συνάφεια με αυτά που έχει ήδη δει, αν έχει λάβει καλή βαθμολογία από ένα γείτονά του. Αυτό δεν εγγυάται την επιτυχία της σύστασης, αλλά μπορεί να βοηθήσει τον χρήστη να διευρύνει τους ορίζοντές του με επιθυμητό για εκείνον τρόπο.

Τα κύρια πλεονεκτήματα αυτών των μεθόδων είναι [25]:

- Απλότητα. Είναι πιο εύκολα κατανοητές διαισθητικά και σχετικά πιο απλές στην υλοποίησή τους.
- Δυνατότητα παροχής δικαιολόγησης για τις προβλέψεις που υπολογίζουν. Μπορούν να επιτρέψουν στο χρήστη να καταλάβει πώς προέκυψαν οι συστάσεις προς αυτόν και λόγω αυτού να χρησιμοποιηθούν σε ένα διαδραστικό σύστημα, όπου οι χρήστες θα μπορούν να επιλέξουν τους γείτονές τους.

- Αποδοτικότητα: Δεν απαιτούν κοστοβόρες φάσεις εκπαίδευσης και οι κοντινότεροι γείτονες μπορούν να προϋπολογιστούν για πιο γρήγορες συστάσεις και να αποθηκευτούν με μικρό κόστος στη μνήμη. Αυτό τους επιτρέπει να χρησιμοποιούνται σε εφαρμογές με εκατομμύρια χρήστες και είδη.

- Σταθερότητα που οφείλεται στο ότι επηρεάζονται ελάχιστα από την αύξηση των χρηστών, των ειδών και των βαθμολογιών στο σύστημα. Για την παροχή συστάσεων σε νέους χρήστες δε χρειάζεται να ξαναϋπολογιστούν οι ομοιότητες μεταξύ ειδών. Όταν ένα νέο είδος λάβει κάποιες βαθμολογίες, οι μόνες ομοιότητες που υπολογίζονται είναι αυτές που το αφορούν.

Η επιλογή ανάμεσα στα δύο είδη συνεργατικής διήθησης βασίζεται στα ακόλουθα κριτήρια[25]:

- Ακρίβεια: Για την επίτευξη ακριβών συστάσεων είναι σημαντική η αναλογία χρηστών και ειδών στο σύστημα. Είναι προτιμότερο να υπολογίζεται η ομοιότητα χρηστών ή ειδών από λιγότερους γείτονες για τους οποίους όμως προκύπτουν περισσότερες κοινές βαθμολογίες καθώς έτσι υπάρχει υψηλότερη εμπιστοσύνη στις παραγόμενες συστάσεις.

- Αποδοτικότητα: Και σε αυτή την περίπτωση έχει σημασία η αναλογία χρηστών και ειδών. Στις περισσότερες περιπτώσεις ο αριθμός των χρηστών υπερβαίνει κατά πολύ αυτόν των ειδών και ο υπολογισμός των γειτονικών ειδών είναι προτιμότερος από άποψης απαιτούμενης μνήμης και χρόνου υπολογισμού των ομοιοτήτων. (Ο χρόνος που απαιτείται για την παραγωγή των συστάσεων είναι ο ίδιος.) Σε μεγαλύτερα συστήματα και λαμβάνοντας υπόψη ότι οι χρήστες πρακτικά βαθμολογούν λίγα αντικείμενα, μπορεί να υπάρξει αποδοτική υλοποίηση αν για κάθε χρήστη αποθηκεύονται μόνο οι καλύτερες ομοιότητες. Με αυτό τον τρόπο δε χρειάζεται να ελέγχονται όλα τα ζεύγη χρηστών ή ειδών.

- Σταθερότητα: Για την παροχή ενός σταθερού συστήματος, παίζει ρόλο η συχνότητα και ο ρυθμός αλλαγής του αριθμού των χρηστών. Ο υπολογισμός ομοιοτήτων με βάσει το είδος είναι προτιμότερος σε συστήματα που ο αριθμός τους αυξάνει πιο αργά σε σχέση με αυτόν των χρηστών και αντίστροφα.

- Δυνατότητα παροχής δικαιολόγησης: Οι βασιζόμενες στα είδη μέθοδοι είναι προτιμότερες στις περιπτώσεις που είναι σημαντικό να παρέχεται στο χρήστη και ο λόγος που το σύστημα του προτείνει ένα είδος, εκτός από τις περιπτώσεις που ο χρήστης γνωρίζει τους άλλους χρήστες (κοινωνικά δίκτυα).

- Δυνατότητα παροχής απρόσμενων, αλλά καλών συστάσεων: Οι μέθοδοι που αξιοποιούν την ομοιότητα μεταξύ των ειδών τείνουν να προτείνουν αποκλειστικά είδη, τα οποία ταιριάζουν με αυτά που έχουν λάβει καλή βαθμολογία

από τον χρήστη. Αντίθετα, οι μέθοδοι που εξετάζουν την ομοιότητα μεταξύ των χρηστών είναι προτιμότερες στην περίπτωση που είναι επιθυμητές και οι απρόσμενες συστάσεις, ιδιαίτερα όταν οι γειτονιές χρηστών που χρησιμοποιούνται είναι μικρές. Αν δύο χρήστες έχουν αρκετές κοινές βαθμολογίες και κάποιος έχει βαθμολογήσει καλά ένα είδος διαφορετικής κατηγορίας από αυτές που ανήκουν τα κοινά τους είδη, τότε είναι πιθανό αυτό το είδος να προταθεί στον άλλο.

Το γεγονός ότι σε πραγματικές εφαρμογές οι χρήστες βαθμολογούν λίγα αντικείμενα καθώς και το ότι ο υπολογισμός ομοιότητας μεταξύ χρηστών προκύπτει από τις κοινές τους βαθμολογίες οδηγεί την προσέγγιση αυτή να αντιμετωπίζει δυο σημαντικές προκλήσεις: τη μειωμένη κάλυψη και την ευαισθησία στα αραιά δεδομένα.

Οι λίγες βαθμολογίες οδηγούν σε λιγότερους γείτονες και έτσι οι μέθοδοι αυτοί δυσκολεύονται να εντοπίσουν ζεύγη χρηστών με παρόμοιες προτιμήσεις, αλλά χωρίς κοινές βαθμολογίες. Αυτό έχει ως αποτέλεσμα λιγότερες συστάσεις, καθώς προτείνονται μόνο είδη που έχουν βαθμολογηθεί από γείτονες, και επομένως μικρότερη κάλυψη του χώρου των ειδών.

Η αραιότητα (sparsity) είναι πρόβλημα που αντιμετωπίζουν τα περισσότερα συστήματα παραγωγής συστάσεων. Όταν δεν υπάρχουν αρκετές βαθμολογίες στο σύστημα, μειώνεται η πιθανότητα δύο χρήστες ή δύο είδη να έχουν κοινές βαθμολογίες. Αυτό επηρεάζει την ακρίβεια των συστάσεων καθώς αυτές προκύπτουν από πολύ μικρές γειτονιές. Σε συνδυασμό και με την προσθήκη νέων χρηστών και ειδών στο σύστημα, προκύπτει το πρόβλημα της κρύας εκκίνησης (cold-start).[2] Το σύστημα μπορεί να οδηγήθει σε κακές συστάσεις καθως μπορεί να θεωρήσει ότι δύο χρήστες έχουν παρόμοιες προτιμήσεις ακόμα αν και οι δύο έχουν βαθμολογήσει ένα και μόνο είδος, αλλά έτυχε να του δώσουν παρόμοια βαθμολογία.

Για την επίλυση των παραπάνω προβλημάτων εφαρμόζονται κάποιες απλές μέθοδοι συμπλήρωσης των βαθμολογιών που λείπουν με κάποια προεπιλεγμένη τιμή. Υπάρχουν διάφορες επιλογές για αυτή την τιμή, όπως οι μέσοι όροι (χρηστών ή ειδών) ή η χρήση πληροφορίας σχετικής με το περιεχόμενο των ειδών. Οι μέθοδοι αυτοί έχουν όμως μειονεκτήματα. Η συμπλήρωση με προεπιλεγμένη τιμή μπορεί να οδηγήσει σε μεροληπτικές συστάσεις και το περιεχόμενο των ειδών μπορεί να μην είναι γνωστό.

Υπάρχουν δύο άλλες αρκετά δημοφιλείς προσεγγίσεις. Η πρώτη βασίζεται στη μείωση της διάστασης των αναπαραστάσεων χρηστών και ειδών, αξιοποιώντας τα πιο σημαντικά χαρακτηριστικά τους. Με αυτόν τον τρόπο μπορεί να ανακαλύψει συσχετίσεις ανάμεσα σε χρήστες και είδη, ακόμα και αν εχουν βαθμολογήσει διαφορετικά είδη ή έχουν βαθμολογηθεί από διαφορετικούς χρήστες. Η δεύτερη εφαρμόζει μεθόδους από τη θεωρία γραφημάτων και μέσω αυτών εντοπίζει μεταβατικές σχέσεις ανάμεσα σε χρήστες και δεδομένα. Έχει επιπλέον τη δυνατότητα

[2] Στο πρόβλημα αυτό θα γίνει αναφορά στην επόμενη ενότητα.

να προτείνει και μη αναμενόμενες, αλλά καλές, συστάσεις.

1.3.2 Συνεργατική Διήθηση Βάσει Μοντέλου

Τα συστήματα αυτής της κατηγορίας δε χρησιμοποιούν απευθείας τις βαθμολογίες για την παραγωγή προβλέψεων, αλλά για να εκπαιδεύσουν ένα μοντέλο πρόβλεψης. Στόχος τους είναι να ανακαλύψουν τα λανθάνοντα χαρακτηριστικά (latent characteristics) χρηστών και ειδών που βρίσκονται πίσω από τις βαθμολογίες. Σε αυτή την κατηγορία ανήκουν δημοφιλείς μέθοδοι που περιλαμβάνουν μοντέλα που προκύπτουν από την παραγοντοποίηση του μητρώου βαθμολογιών χρηστών-ειδών. Είναι γνωστές ως μέθοδοι που βασίζονται στον αλγόριθμο SVD (SVD-based) και διακρίνονται για την ακρίβεια και την ικανότητά τους να ανταποκρίνονται καλά στην αύξηση των αριθμών χρηστών, ειδών και βαθμολογιών. Επιπλέον, προσφέρουν ένα μοντέλο που μπορεί να αποθηκευτεί αποδοτικά και μπορεί να εκπαιδευτεί σχετικά εύκολα.

Στη βασική τους μορφή, μοντελοποιούν τις αλληλεπιδράσεις χρηστών-ειδών μετασχηματίζοντας χρήστες και είδη στον ίδιο χώρο λανθάνοντων παραγόντων (latent factors). Κάθε χρήστης συσχετίζεται με ένα διάνυσμα του οποίου τα στοιχεία καταγράφουν το ενδιαφέρον του για τους διάφορους παράγοντες. Αντίστοιχα, κάθε είδος συσχετίζεται με ένα διάνυσμα που απεικονίζει το κατά πόσο έχει αυτά τα χαρακτηριστικά. Το εσωτερικό τους γινόμενο υπολογίζει το ενδιαφέρον του χρήστη για τα χαρακτηριστικά του είδους. Στη συνέχεια το μοντέλο εκπαιδεύεται, χρησιμοποιώντας τα διαθέσιμα δεδομένα, για να προβλέψει τις βαθμολογίες χρηστών για είδη που δεν έχουν δει. Τα συστήματα αυτά διευρύνουν την κατηγοριοποίηση των ειδών καθώς είναι σε θέση να εντοπίσουν το ενδιαφέρον ενός χρήστη για είδη που έχουν χαρακτηριστικά που εξάγονται αυτόματα από το σύστημα και επομένως να του προτείνουν πιο στοχευμένα είδη. Αλγόριθμοι αυτού του είδους, όπως ο SVD++ [17] μπορούν να χρησιμοποιήσουν και άλλα στοιχεία του ιστορικού του χρήστη πέρα από τις βαθμολογίες, αυξάνοντας την ακρίβεια των προβλέψεων. Μέθοδοι αυτής της κατηγορίας μπορούν ακόμα να λάβουν υπόψη τους τις αλλαγές στις προτιμήσεις των χρηστών και στα χαρακτηριστικά των ειδών στη διάρκεια του χρόνου, βελτιώνοντας και άλλο την ποιότητα των προβλέψεων. [25]

Άλλη μέθοδος συνεργατικής δίηθησης βάσει μοντέλου είναι η Γκαουσιάνη Πιθανοτική Λανθάνουσα Σημασιολογική Ανάλυση (Gaussian Probabilistic Latent Semantic Analysis)[13]. Βάσει αυτής, οι βαθμολογίες των χρηστών μοντελοποιούνται ως ένας συνδυασμός κοινοτήτων χρηστών ή ομάδων ειδών στις οποίες οι χρήστες ή τα είδη συμμετέχουν πιθανοτικά. Η μέθοδος αυτή πετυχαίνει αυξημένη ακρίβεια πρόβλεψης, συμπιέζει τα δεδομένα σε ένα συμπαγές μοντέλο που εντοπίζει αυτόματα τις κοινότητες χρηστών και τις ομάδες ειδών και δίνει τη δυνατότητα υπολογισμού των προβλέψεων προτίμησης σε σταθερό χρόνο.

1.4 Το πρόβλημα της Κρύας Εκκίνησης

Το πρόβλημα της κρύας εκκίνησης αναφέρεται στην συμπεριφορά του συστήματος στην εισαγωγή νέων χρηστών και ειδών και στη δυσκολία παραγωγής αξιόπιστων συστάσεων λόγω της αρχικής έλλειψης βαθμολογιών. Είναι πρόβλημα που αντιμετωπίζουν όλα τα συστήματα παραγωγής συστάσεων, τόσο αυτά που βασίζονται στο περιεχομένο όσο και αυτά που εφαρμόζουν συνεργατική διήθηση. Για τα δεύτερα είναι πιο σοβαρό πρόβλημα καθώς βασίζονται αποκλειστικά στις βαθμολογίες των χρηστών. Μπορεί να θεωρηθεί και ως πρόβλημα μειωμένης κάλυψης του χώρου των ειδών.[25] Διακρίνεται κυρίως σε τρεις κατηγορίες [24]:

- Προβλημα Νέων Χρηστών (New Users Problem): Το πρόβλημα αυτό εμφανίζεται με την εισαγωγή νέων χρηστών στο σύστημα, οι οποίοι δε μπορούν να λάβουν προσωποποιημένες συστάσεις, αφού έχουν ελάχιστες βαθμολογίες.

- Πρόβλημα Νέων Ειδών (New Items Problem): Επειδή τα νέα είδη που εισάγονται στο σύστημα εκ των πραγμάτων δεν έχουν βαθμολογηθεί αρκετά, δε μπορούν να συσχετιστούν με τα υπόλοιπα και επομένως να συμπεριληφθούν στις συστάσεις προς τους χρήστες.

- Πρόβλημα Νέας Κοινότητας (New Community Problem): Στην αρχή της λειτουργίας ενός συστήματος, οι βαθμολογίες είναι αναγκαστικά λίγες. Αυτό οδηγεί σε αραιά σύνολα δεδομένων και δεν επιτρέπει στα συστήματα συνεργατικής διήθησης να εντοπίσουν τις απαραίτητες συσχετίσεις ανάμεσα σε χρήστες και είδη.

Απλές προσεγγίσεις για την αντιμετώπιση αυτού του προβλήματος απαιτούν από το χρήστη με την εισαγωγή του στο σύστημα να βαθμολογήσει κάποια είδη ή να απαντήσει κάποιες δημογραφικές ερωτήσεις. Αυτές οι προσεγγίσεις έχουν τα μειονεκτήματα ότι μπορεί ο χρήστης να μη θέλει να αφιερώσει χρόνο για να δώσει πληροφορίες στο σύστημα ή να μην επιθυμεί να δώσει δεδομένα του, έστω και δημογραφικής φύσης. Ακόμα κάποιοι χρήστες μπορεί να δυσαρεστηθούν ή να προσβληθούν αν λάβουν συστάσεις από το σύστημα που προκύπτουν από τη χρήση στερεοτύπων σε σχέση με την ηλικία, το φύλο, την καταγωγή τους κτλ.[25] Επίσης, δεν είναι εύκολο για το σύστημα να επιλέξει ποια θα είναι τα πρώτα είδη για τα οποία θα ζητήσει τη γνώμη του χρήστη. Μια άλλη επιλογή για το σύστημα, όσον αφορά τους νέους χρήστες, είναι να προτείνει σε αυτούς δημοφιλή είδη. Σε αυτή την περίπτωση όμως είναι πολύ πιθανό ο χρήστης ήδη να τα γνωρίζει και να θεωρήσει ότι το σύστημα δεν καλύπτει τις ανάγκες του. Άλλα συστήματα προσπαθούν να λάβουν πληροφορίες για το ποιους άλλους χρήστες τους μπορεί να γνωρίζει, ώστε του προτείνουν είδη που θα πρότειναν σε εκείνους ή είδη για τα οποία εκείνοι έχουν δείξει προτίμηση.[3] Αυτές τις πληροφορίες μπορούν να τις λάβουν, αν παρέχουν στο χρήστη τη δυνατότητα να συνδέσει το προφίλ του

[3] https://www.coursera.org/learn/recommender-systems

με κάποιο προφίλ που μπορεί να έχει σε κάποιο άλλο σύστημα, όπως κάποιο κοινωνικό δίκτυο. Έτσι μπορούν να αποκτήσουν πρόσβαση στη λίστα των επαφών του και να ελέγξουν αν κάποια από αυτές χρησιμοποιεί το σύστημά τους.

Μια άλλη μέθοδος, η οποία προτείνεται στο [19], θεωρεί ότι ο νέος χρήστης ταιριάζει με κάποιον από τους ήδη υπάρχοντες, χωρίς φυσικά να μπορεί να γνωρίζει με ποιον. Στη συνέχεια προσπαθεί να παράξει συστάσεις, οι οποίες είναι ικανοποιητικές τόσο για το νέο χρήστη όσο και για τους υπόλοιπους, υποθέτοντας ότι αν οι συστάσεις είναι καλές για όλους τους υπόλοιπους χρήστες, θα είναι και για εκείνον. Αρχικά δίνει ένα μικρό βάρος στο νέο χρήστη και περισσότερο στους άλλους χρήστες. Σταδιακά και όσο το σύστημα μαθαίνει τις προτιμήσεις του νέου χρήστη το βάρος που του δίνεται μεγαλώνει (και ταυτόχρονα μειώνεται αυτό των άλλων χρηστών και περισσότερο αυτών που φαίνεται να έχουν διαφορετικές προτιμήσεις από εκείνον) μέχρι που κυριαρχεί πλήρως στην επιλογή των συστάσεων προς αυτόν.

Όσον αφορά τα νέα είδη, υπάρχουν κάποιες απλές μέθοδοι, όπως το να προτείνονται τυχαία σε κάποιους χρήστες, ώστε το σύστημα να καταγράψει την αντίδρασή τους σε αυτά και να υπολογίσει την ομοιότητά τους με άλλα είδη. Άλλη επιλογή είναι να τα προτείνει είτε σε χρήστες που ενδιαφέρονται να ανακαλύψουν νέα είδη είτε σε χρήστες που έχουν βαθμολογήσει αρκετά είδη, ώστε να μπορέσουν να τα συσχετίσουν με αυτά. Μπορεί επίσης να χρησιμοποιηθεί, αν υπάρχει, γνώση για το περιεχόμενό τους. Ακόμα μπορεί να γίνει χρήση ετικετών (tags).

Η περίπτωση της νέας κοινότητας είναι η πιο δύσκολη από τις τρεις. Μια βολική επιλογή για το νέο σύστημα είναι, αν έχει τη δυνατότητα, να χρησιμοποιήσει δεδομένα από κάποια άλλη πηγή. Μπορεί ακόμα στην αρχή της λειτουργίας του και μέχρι να αποκτήσει αρκετές (είτε άμεσες είτε έμμεσες) πληροφορίες από τους χρήστες να λειτουργεί ως ένα μη προσωποποιημένο σύστημα παραγωγής συστάσεων.[4]

1.5 Συνεισφορά και Οργάνωση της Διπλωματικής

Σε αυτό το κεφάλαιο έγινε μια επισκόπηση των διαφόρων προσεγγίσεων που ακολουθούν τα συστήματα παραγωγής συστάσεων. Περισσότερη έμφαση δόθηκε στην παρουσίαση των βασικότερων τεχνικών συνεργατικής δίηθησης. Τέλος, έγινε αναφορά στο πρόβλημα της κρύας εκκίνησης και σε κάποιες από τις τεχνικές για την αντιμετώπισή του.

Στο 2ο κεφάλαιο παρουσιάζεται ο αλγόριθμος Hierarchical Itemspace Rank (HIR), αναλύεται το μοντέλο που προτείνει και αναφέρεται η πειραματική του αξιολόγηση, η οποία αποδεικνύει την υπεροχή του στο πρόβλημα της κρύας εκκίνησης. Η επιτυχία του οφείλεται στην εκμετάλλευση της εγγενούς ιεραρχικής δομής του χώρου των ειδών.

[4] https://www.coursera.org/learn/recommender-systems

Στο 3ο κεφάλαιο γίνεται μια επισκόπηση του LensKit, της στρατηγικής που ακολουθεί και των πλεονεκτημάτων που προσφέρει. Το LensKit αναπτύχθηκε με γνώμονα τη διευκόλυνση όσων επιθυμούν να χρησιμοποιήσουν ή να υλοποιήσουν αλγορίθμους παραγωγής συστάσεων. Παρέχει ακόμα τα μέσα για την πειραματική τους αξιολόγηση.

Στο 4ο κεφάλαιο αναλύεται η υλοποίηση του αλγορίθμου HIR στο LensKit, η οποία πραγματοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Παρουσιάζεται ακόμα το Demo, το οποίο υλοποιήθηκε και κάνει χρήση του HIR καθώς και τα σεναρία που γράφτηκαν για τον έλεγχο της υλοποίησης.

Στο 5ο κεφάλαιο περιέχονται τα συμπεράσματα που προέκυψαν κατά την εκπόνηση της παρούσας διπλωματικής εργασίας καθώς και το πώς θα μπορούσε να χρησιμοποιήθεί πιθανά στο μέλλον η υλοποίηση που έγινε.

Στο 6ο κεφάλαιο παρατίθεται το σύνολο του κώδικα της υλοποίησης, του Demo, των σεναρίων ελέγχου και του βοηθητικού κώδικα που γράφτηκε για το χειρισμό των συνόλων δεδομένων και για την παραγωγή των απαιτούμενων δεδομένων για τα σενάρια ελέγχου.

Στο 7ο και το 8ο κεφάλαιο καταγράφονται αντίστοιχα η ορολογία και τα αρκτικόλεξα που χρησιμοποιήθηκαν στο κείμενο της διπλωματικής.

Τέλος, παρατίθεται η σχετική βιβλιογραφία.

Κεφάλαιο 2

Ο Αλγόριθμος HIR

Η αραιότητα (sparsity) είναι εγγενές χαρακτηριστικό των συστημάτων παραγωγής συστάσεων και αποτελεί ένα από τα πιο δύσκολα ζητήματα που έχουν να αντιμετωπίσουν οι αλγόριθμοι συνεργατικής δίηθησης. Σε αυτό ακριβώς το πρόβλημα, ακόμα και στην ακραία του μορφή που είναι το πρόβλημα της κρύας εκκίνησης, απαντάει ο Hierarchical Itemspace Rank (HIR). Ο HIR εκμεταλλεύεται την ιεραρχική δομή του χώρου των ειδών, ώστε να ξεπεράσει τους περιορισμούς στην ποιοτική παραγωγή συστάσεων[24].[1]

2.1 Εισαγωγή

Το έναυσμα για τον αλγόριθμο HIR προέκυψε από τα αποτελέσματα ενός καινοτόμου πλαισίου εργασίας στον Παγκόσμιο Ιστό [23]. Το πλαίσιο αυτό κατέδειξε τα οφέλη της απεικόνισης των σχέσεων γειτνίασης μεταξύ των ιστοσελίδων που προκύπτουν από την εγγενή ιεραρχική δομή του Παγκόσμιου Ιστού και της εκμετάλλευσής τους για την καλύτερη κατάταξη των ιστοσελίδων που επιστρέφονται από μία μηχανή αναζήτησης.

Ο αλγόριθμος HIR εκμεταλλεύεται την ιεραρχική δομή που υπάρχει εκ φύσεως στο χώρο των ειδών, ώστε να χαρακτηρίσει τις σχέσεις μεταξύ των ειδών σε μακροσκοπικό επίπεδο. Για το σκοπό αυτό αναλύει το χώρο των ειδών σε σύνολα στενά συνδεδεμένων στοιχείων και χρησιμοποιεί αυτή την ανάλυση για να εκμεταλλευτεί τις σχέσεις γειτνίασης που προκύπτουν ανάμεσα στα είδη.

Κεντρική του ιδέα είναι ο συνδυασμός των άμεσων σχέσεων που προκύπτουν από τις αλληλεπιδράσεις των χρηστών με τα είδη και των έμμεσων που προκύπτουν από την κατηγοριοποίηση των ειδών με στόχο την αντιμετώπιση του προβλήματος της αραιότητας και τη βελτίωση της ποιότητας των συστάσεων.

[1] Κύρια πηγή του παρόντος κεφαλαίου ειναι η δημοσίευση [24].

2.2 Το πλαίσιο εργασίας

Στην ενότητα αυτή παρουσιάζεται το πλαίσιο εργασίας του αλγορίθμου.

2.2.1 Σημειογραφία

- Όλα τα διανύσματα-στήλες αναπαριστώνται με έντονα πεζά γράμματα.
- Όλα τα μητρώα με έντονα κεφαλαία γράμματα.
- $\mathbf{c}_j^{\mathrm{T}}$ είναι η j στήλη του μητρώου $\mathbf{C}$ και C_{ij} είναι το στοιχείο στη γραμμή i και στη στήλη j του μητρώου $\mathbf{C}$.
- Με κεφαλαία καλλιγραφικά γράμματα ορίζονται τα σύνολα και
- το $\triangleq$ χρησιμοποιείται στους ορισμούς.

2.2.2 Ορισμός του μοντέλου

Ορίζονται τα σύνολα:

- Χρηστών: $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$
- Ειδών: $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$
- Βαθμολογιών: $\mathcal{R}$, το οποίο αποτελείται από πλειάδες $t_{ij} = (u_i, v_j, r_{ij})$, όπου r_{ij} είναι η βαθμολογία του χρήστη u_i για το είδος v_j, η οποία μπορεί να αντιστοιχεί είτε σε ένα θετικό αριθμό είτε σε μια δυαδική μορφή που θα παρουσιάζει την αλληλεπίδραση ή μη του χρήστη με ένα είδος.
- Διαμέρισης του συνόλου $\mathcal{R}$: $\{\mathcal{L}, \mathcal{T}\}$, με το $\mathcal{L}$ να αποτελεί το σύνολο εκπαίδευσης και το $\mathcal{T}$ το συνολο ελέγχου. Το $\mathcal{L}_i$ περιέχει τα είδη που έχει βαθμολογήσει ο u_i και ανήκουν στο σύνολο εκπαίδευσης. Το $\mathcal{T}_i$ περιέχει τα είδη που έχει βαθμολογήσει ο u_i και ανήκουν στο σύνολο ελέγχου:

$$\mathcal{L}_i \triangleq \{v_k : t_{ik} \in \mathcal{L}\} \text{ και } \mathcal{T}_i \triangleq \{v_l : t_{il} \in \mathcal{T}\}$$

Κάθε χρήστης u_i συσχετίζεται με το διάνυσμα προτίμησής του, $\boldsymbol{\omega}^i$, του οποίου τα μη μηδενικά στοιχεία αντιστοιχούν στις βαθμολογίες που έχει δώσει ο χρήστης και ανήκουν στο $\mathcal{L}$ και για τον οποίο ισχύει $\mathcal{L}_i \neq \emptyset$. Το διάνυσμα προτίμησης είναι κανονικοποιημένο, ώστε τα στοιχεία του να αθροίζουν στη μονάδα.

$$\boldsymbol{\omega}^i = [\omega_1^i, \omega_2^i, \ldots, \omega_m^i]$$

Στόχος της μεθόδου που προτείνει ο αλγόριθμος HIR είναι η αντιστοίχιση του σε μία προσωποποιημένη κατανομή πάνω στο χώρο των ειδών.

Επιπλέον ορίζεται μια οικογένεια μη κενών συνόλων πάνω στο χώρο ειδών $\mathcal{V}$ με βάση κάποιο κριτήριο κατηγοριοποίησης τους:

$$\mathcal{D} \triangleq \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}, \text{ με } \mathcal{V} = \bigcup_{k=1}^{K} \mathcal{D}_k$$

Τέλος, ορίζεται το $\mathcal{G}_v$, το οποίο αποτελεί την ένωση των συνόλων που περιέχουν το v και N_v το πλήθος αυτών:

$$\mathcal{G}_v \triangleq \bigcup_{v \in \mathcal{D}_k} \mathcal{D}_k$$

Μητρώο Άμεσης Συσχέτισης $\mathbf{C}$

Το μητρώο άμεσης συσχέτισης $\mathbf{C}$ στοχεύει στο να απεικονίσει τις άμεσες σχέσεις που προκύπτουν ανάμεσα στα στοιχεία του συνόλου των ειδών $\mathcal{V}$. Κάθε στοιχείο του $\mathcal{V}$ συσχετίζεται με μία διακριτή κατανομή πάνω σε αυτό, η οποία ποσοτικοποιεί τις ομοιότητες των στοιχείων του.

$$\mathbf{c}_v^{\mathrm{T}} = [c_1, c_2, \ldots, c_m]$$

Για να οριστεί το μητρώο $\mathbf{C}$, χρειάζεται να οριστεί η οικογένεια συνόλων $\mathcal{U}_{ij} \subseteq \mathcal{U}$, που αντιπροσωπεύει το σύνολο των χρηστών που έχουν βαθμολογήσει και το v_i και το v_j.

$$\mathcal{U}_{ij} \triangleq \begin{cases} \{u_k : (v_i \in \mathcal{L}_k) \wedge (v_j \in \mathcal{L}_k)\} & i \neq j \\ \emptyset & i = j \end{cases}$$

Ορίζεται τώρα το μητρώο $\mathbf{Q}$, με $Q_{ij} \triangleq |\mathcal{U}_{ij}|$. Το μητρώο αυτό είναι συμμετρικό και η διαγώνιός του ειναι μηδενική. Η στοχαστική του έκδοση $\hat{\mathbf{Q}}$ αντιστοιχεί στο γράφημα συσχέτισης των ειδών, όπως ορίζεται στο [12], όπου κάθε ακμή του γραφήματος έχει ως βάρος το αντίστοιχο στοιχείο του μητρώου. Το μητρώο αυτό σε αντίθεση με το $\mathbf{Q}$ δεν είναι συμμετρικό. Το μητρώο $\mathbf{C}$ ορίζεται ως:

$$\mathbf{C} \triangleq \hat{\mathbf{Q}} + \tfrac{1}{m}\mathbf{a}\mathbf{e}^{\mathrm{T}}$$

όπου το διάνυσμα $\mathbf{a} \in \Re^m$ έχει μονάδες στα στοιχεία που αντιστοιχούν στις μηδενικές γραμμές του $\mathbf{Q}$ και το $\mathbf{e}^{\mathrm{T}} \in \Re^m$ είναι το μοναδιαίο διάνυσμα. Το μητρώο που προστίθεται στο $\hat{\mathbf{Q}}$ στοχεύει στο να αντικαταστήσει τις μηδενικές γραμμές του με μια ομοιόμορφη κατανομή στο χώρο των ειδών. Αυτή η μετατροπή συμβάλλει στο να προκύψει κάποια πληροφορία για ζεύγη ειδών όπου τα σύνολα των χρηστών που τις έχουν βαθμολογήσει είναι ξένα μεταξύ τους. Τέτοια σύνολα είναι πιο πιθανό να βρεθούν σε συστήματα συστάσεων που αντιμετωπίσουν το πρόβλημα της κρύας εκκίνησης.

Μητρώο Γειτνίασης $\mathbf{D}$

Το μητρώο $\mathbf{D}$ στοχεύει στο να απεικονίσει τις σχέσεις ανάμεσα στο χώρο των ειδών όπως προκύπτουν από την ιεραρχική του δομή. Αυτή η δομή προκύπτει από το γεγονός ότι η έκφραση προτίμησης ενός χρήστη για ένα είδος υποδηλώνει

το ενδιαφέρον του για τις κατηγορίες στις οποίες ανήκει, άρα και για τα είδη που ανήκουν σε αυτές. Η απεικόνιση αυτών των σχέσεων μπορεί να φανεί πολύ χρήσιμη για την αντιμετώπιση του προβλήματος της αραιότητας.

Με βάση τα παραπάνω, κάθε γραμμή του $\mathbf{D}$ συσχετίζεται με ένα διάνυσμα πιθανότητας $\mathbf{d}_v^{\mathrm{T}}$, το οποίο κατανέμει ομοιόμορφα τη μάζα του στα N_v διαφορετικά σύνολα του $\mathcal{D}$ που αποτελούν το $\mathcal{G}_v$ και στη συνέχεια στα είδη που αποτελούν το κάθε ένα από αυτά. Κάθε στοιχείο του $\mathbf{D}$, το οποίο συνδέει τα είδη v_i και v_j ορίζεται ως:

$$D_{ij} \triangleq \sum_{\mathcal{D}_k \in \mathcal{G}_{v_i}, v_j \in \mathcal{D}_k} \frac{1}{\mathrm{N}_{v_i} |\mathcal{D}_k|}$$

Τόσο από τον ορισμό του, όσο και από τον ορισμό της οικογένειας συνόλων $\mathcal{D}$, προκύπτει ότι το $\mathbf{D}$ είναι στοχαστικό κατά γραμμές.

2.2.3 Ο Αλγόριθμος Ιεραρχικής Κατάταξης βάσει του χώρου των ειδών

Με βάση τα παραπάνω, μπορεί να οριστεί το προσωποποιημένο διάνυσμα κατάταξης του HIR ως η κατανομή πιθανότητας πάνω στο χώρο των ειδών, όπως παράγεται από τον αλγόριθμο 1:

Αλγόριθμος 1 Ιεραρχική Κατάταξη Βάσει του Χώρου των Ειδών

Είσοδος: Μητρώα $\mathbf{C}$, $\mathbf{D} \in \Re^{m \times m}$, παράμετροι α, $\beta : \alpha, \beta > 0$, $\alpha + \beta < 1$ και το προσωποποιημένο διάνυσμα προτίμησης $\boldsymbol{\omega} \in \Re^m$

Έξοδος: Το διάνυσμα κατάταξης για τον χρήστη, $\boldsymbol{\pi} \in \Re^m$

1: $\boldsymbol{\pi}^{\mathrm{T}} \leftarrow (1 - \alpha - \beta)\boldsymbol{\omega}^{\mathrm{T}}$
2: **for all** $\omega_j \neq 0$ **do**
3: $\quad \boldsymbol{\pi}^{\mathrm{T}} \leftarrow \boldsymbol{\pi}^{\mathrm{T}} + \omega_j(\alpha \mathbf{c}_j^{\mathrm{T}} + \beta \mathbf{d}_j^{\mathrm{T}})$
4: **end for**
5: **return** $\boldsymbol{\pi}$

Θεώρημα 1. *Για κάθε διάνυσμα προτίμησης $\boldsymbol{\omega}$, το προσωποποιημένο διανυσμα $\boldsymbol{\pi}$, το οποίο παράγεται από τον αλγόριθμο HIR είναι ένα καλώς ορισμένο, κανονικοποιημένο διάνυσμα συστάσεων, το οποίο απεικονίζει μία κατανομή πιθανότητας στο σύνολο των ειδών $\mathcal{V}$.*

Απόδειξη. Για κάθε διάνυσμα προτίμησης $\boldsymbol{\omega}$ (το οποίο είναι από τον ορισμό του μη αρνητικό) και για $\alpha, \beta > 0$, $\alpha + \beta < 1$, το $\boldsymbol{\pi}$ είναι μη αρνητικό διάνυσμα. Επομένως, αρκεί να δειχθεί ότι $\boldsymbol{\pi}^{\mathrm{T}}\mathbf{e} = 1$:

$$\boldsymbol{\pi}^{\mathrm{T}}\mathbf{e} = ((1 - \alpha - \beta)\boldsymbol{\omega}^{\mathrm{T}} + \alpha \sum_{j:\omega_j \neq 0} \omega_j \mathbf{c}_j^{\mathrm{T}} + \beta \sum_{j:\omega_j \neq 0} \omega_j \mathbf{d}_j^{\mathrm{T}})\mathbf{e}$$

Δεδομένου ότι τα στοιχεία του διανύσματος $\boldsymbol{\omega}$ είναι κανονικοποιημένα να αθροίζουν στη μονάδα και τα μητρώα $\mathbf{C}$ και $\mathbf{D}$ είναι στοχαστικά κατά γραμμές, προκύπτει ότι:

$$\boldsymbol{\pi}^{\mathrm{T}}\mathbf{e} = (1-\alpha-\beta) + \alpha \sum_{j:\omega_j \neq 0} \omega_j + \beta \sum_{j:\omega_j \neq 0} \omega_j = (1-\alpha-\beta) + \alpha + \beta = 1$$

□

2.2.4 Ζητήματα Απαιτούμενης Μνήμης

Το μητρώο $\mathbf{C}$ είναι εγγενώς αραιό και κλιμακώνεται πολύ καλά με την αύξηση του αριθμού των χρηστών. Η αύξηση του αριθμού των χρηστών οδηγεί μόνο σε αύξηση του αριθμού των μη μηδενικών στοιχείων του, αφού η διάστασή του εξαρτάται μόνο απο το πλήθος των ειδών. Σε πραγματικές εφαρμογές αυτό το πλήθος αυξάνει πολυ αργά. [12]

Το μητρώο $\mathbf{D}$ είναι από τον ορισμό του χαμηλής τάξης για $K < m$. Το $\mathbf{D}$ μπορεί να παραγοντοποιηθεί και να επιτευχθεί έτσι αποδοτικότερη αποθήκευσή του και μείωση του υπολογιστικού κοστους. Για το σκοπό αυτό, ορίζεται το μητρώο $\mathbf{A} \in \Re^{m \times K}$, ώστε:

$$A_{ik} \triangleq \begin{cases} 1 & \text{αν } v_i \in \mathcal{D}_k \\ 0 & \text{αλλιώς} \end{cases}$$

Το $\mathbf{D}$ μπορεί να γραφεί ως το γινόμενο των στοχαστικών κατά γραμμές εκδόσεων του $\mathbf{A}$ και του αναστρόφου του αντίστοιχα.

$$\mathbf{D} = \mathbf{XY},\ \mathbf{X} \in \Re^{m \times K},\ \mathbf{Y} \in \Re^{K \times m},\ \text{με}$$
$$\mathbf{X} \triangleq \mathbf{S}^{-1}\mathbf{A} \text{ και}$$
$$\mathbf{Y} \triangleq \mathbf{T}^{-1}\mathbf{A}^{\mathrm{T}}$$

όπου τα μητρώα $\mathbf{S} \triangleq \mathrm{diag}(\mathbf{Ae})$ και $\mathbf{T} \triangleq \mathrm{diag}(\mathbf{A}^{\mathrm{T}}\mathbf{e})$ είναι διαγώνια.

Λήμμα 1. *Τα μητρώα $\mathbf{X}$ και $\mathbf{Y}$ είναι καλώς ορισμένα για οποιαδήποτε παραγοντοποίηση $\mathcal{D}$ ικανοποιώντας τον ορισμό του.*

Απόδειξη. Αρκεί να δειχθεί ότι τα $\mathbf{S}$ και $\mathbf{T}$ είναι αντιστρέψιμα για οποιοδήποτε μητρώο $\mathbf{A}$. Από τον ορισμό του $\mathcal{D}$ προκύπτει ότι κάθε γραμμή και κάθε στήλη του $\mathbf{A}$ αντιστοιχεί σε ένα μη μηδενικό διάνυσμα στον $\Re^K$ και $\Re^m$ αντίστοιχα.

Η ύπαρξη μηδενικής γραμμής στο μητρώο $\mathbf{A}$, θα σήμαινε ότι το σύνολο $\mathcal{V}$ δεν αποτελεί την ένωση της οικογένειας συνόλων $\mathcal{D}_k$, άτοπο από τον ορισμό του $\mathcal{D}$.

Η ύπαρξη μηδενικής στήλης στο μητρώο $\mathbf{A}$, θα προέκυπτε μόνο αν κάποιο από τα $\mathcal{D}_k$ ήταν κενό, το οποίο αντιτίθεται στον ορισμό τους και δε θα είχε νόημα βάσει του ορισμού του μοντέλου.

Τα παραπάνω οδηγούν στο συμπέρασμα ότι τα διανύσματα $\mathbf{Ae}$, $\mathbf{A}^{\mathrm{T}}\mathbf{e}$ είναι αυστηρά θετικά, διασφαλίζοντας την αντιστρεψιμότητα των μητρώων $\mathbf{S}$, $\mathbf{T}$. □

2.2.5 Υπολογιστικά Ζητήματα

Το κόστος εκτέλεσης του αλγορίθμου 1 είναι μικρό καθώς η δομή επανάληψης απαιτεί $\mathcal{O}(|\mathcal{V}|)$ πράξεις και εκτελείται $\frac{|\mathcal{L}_i|}{m}$ φορές καθώς οι χρήστες αλληλεπιδρούν με ένα πολύ μικρό μέρος των διαθέσιμων ειδών.

Η παραγοντοποίηση του $\mathbf{D}$ ως γινόμενο δύο εξαιρετικά αραιών μητρώων, καταργώντας την ανάγκη απευθείας υπολογισμού του, καθώς και το χαμηλής πυκνότητας μητρώο βαθμολογιών, μειώνουν και άλλο το υπολογιστικό κόστος καθώς επιτρέπουν το μαζικό υπολογισμό των προσωποποιημένων διανυσμάτων κατάταξης που υπολογίζει ο αλγόριθμος 1.

Για τον υπολογισμό αυτό χρειάζεται να οριστεί ένα μητρώο $\mathbf{\Omega} \in \Re^{n \times m}$, τέτοιο ώστε:

$$\mathbf{\Omega} \triangleq \begin{bmatrix} (\boldsymbol{\omega^1})^{\mathrm{T}} \\ (\boldsymbol{\omega^2})^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{\omega^n})^{\mathrm{T}} \end{bmatrix}$$

Ο μαζικός αυτός υπολογισμός του HIR υπολογίζεται με τον Αλγόριθμο 2, με τις γραμμές του μητρώου $\mathbf{\Pi} \in \Re^{n \times m}$, να περιέχουν τα διανύσματα συστάσεων για κάθε χρήστη του συνόλου $\mathcal{U}$. Ο υπολογισμός αυτός επιτρέπει την εκμετάλλευση εξαιρετικά βελτιστοποιημένων αραιών BLAS3 συνόλων χαμηλού επιπέδου υπορουτίνων του πυρήνα.

Αλγόριθμος 2 Μαζικός Υπολογισμός του HIR

Είσοδος: Μητρώα $\mathbf{C}$, $\mathbf{X}$, $\mathbf{Y}$, παράμετροι $\alpha,\ \beta\ :\ \alpha, \beta > 0,\ \alpha + \beta < 1$ και το προσωποποιημένο μητρώο προτίμησης $\mathbf{\Omega}$
Έξοδος: Το μητρώο κατάταξης για τους χρήστες, $\mathbf{\Pi} \in \Re^{n \times m}$

1: $\mathbf{\Pi} \leftarrow (1 - \alpha - \beta)\mathbf{\Omega}$
2: $\mathbf{\Pi} \leftarrow \mathbf{\Pi} + \alpha(\mathbf{\Omega C}) + \beta((\mathbf{\Omega X})\mathbf{Y})$
3: **return** $\mathbf{\Pi}$

2.3 Πειραματική Αξιολόγηση

Η πειραματική αξιολόγηση[24] του HIR, έγινε στους τομείς των συστάσεων ταινιών και μουσικής.

Για τον πρώτο τομέα, τα πειράματα έγιναν στα σύνολα δεδομένων MovieLens100K και MovieLens1M, με το διαχωρισμό των ταινιών σε κατηγορίες να αποτελεί το κριτήριο για τον ορισμό του μητρώου γειτνίασης. Τα σύνολα αυτά χρησιμοποιούνται ευρέως για τον έλεγχο απόδοσης των συστημάτων παραγωγής συστάσεων. Τα σύνολα αυτά περιλαμβάνουν:

Σύνολα Δεδομένων	**Χρήστες**	**Είδη**	**Βαθμολογίες**
MovieLens100K	943	1.682	100.000
MovieLens1M	6.040	3.883	1.000.209

Για το δεύτερο τομέα, τα πειράματα έγιναν στο σύνολο δεδομένων Yahoo!R2Music με κριτήριο κατηγοριοποίησης τους καλλιτέχνες που ερμηνεύουν τα τραγούδια. Το σύνολο αυτό περιλαμβάνει:

Χρήστες	**Είδη**	**Βαθμολογίες**
1.823.179	136.736	717.872.016

Η σύγκριση του HIR έγινε με τους ακόλουθους αλγορίθμους παραγωγής συστάσεων που βασίζονται σε κατατάξεις:

- L[8] και Katz [14], οι οποίοι βασίζονται σε ομοιότητα κόμβων
- First Passage Time (FP)[7] και Matrix Forest Algorithm (FMA)[3] που ακολουθούν την προσέγγιση του τυχαίου περιπάτου
- ItemRank [12] και
- PureSVD [4]

Οι δύο τελευταίοι δε μπορούν να επωφεληθούν από την ιεραρχική δομή του χώρου των ειδών. Η υλοποίηση όλων των αλγορίθμων έγινε σε MATLAB.

Από τους παραπάνω αλγορίθμους μόνο ο ItemRank και ο HIR εξαρτώνται αποκλειστικά από το μέγεθος του χώρου των ειδών, του οποίου η διάσταση σε πραγματικές εφαρμογές αυξάνεται πολύ αργά. Σε όλα τα παραπάνω σύνολα δεδομένων ο HIR ήταν 10-15 φορές πιο γρήγορος.

Για τα πειράματα χρησιμοποιήθηκαν οι μετρικές:

- Spearman's ρ [20] και Kendall's τ [15], που αποτελούν μη παραμετρικές μετρικές για τη συσχέτιση ανάμεσα σε λίστες κατάταξης,
- Degree of Agreement (DOA) [26, 12] (Βαθμός Συμφωνίας), η οποία εξετάζει κατά πόσο η επιστρεφόμενη λίστα κατάταξης είναι ικανοποιητική για τον χρήστη. Αυτό το πετυχαίνει μετρώντας το ποσοστό των ζευγών της λίστας που έχουν σωστή διάταξη, σε σχέση με το σύνολο τους και
- Normalized Distance-based Performance Measure [29, 9] (Μετρική Κανονικοποιημένης Απόδοσης Βάσει Απόστασης): Υπολογίζει την (κανονικοποιημένη) απόσταση ανάμεσα στη λίστα κατάταξης του χρήστη και σε αυτή που επιστρέφει το σύστημα ως προτεινόμενη. Η επιστρεφόμενη τιμή για την ελάχιστη απόσταση (πλήρης ταύτιση) είναι 0 και η μέγιστη 1.

Η σύγκριση των αλγορίθμων έγινε πάνω στα προβλήματα της Νέας Κοινότητας, των Νέων Χρηστών και των Νέων Ειδών που αποτελούν τις τρεις εκφάνσεις του προβλήματος της κρύας εκκίνησης.

Για τη μοντελοποίηση του προβλήματος της Νέας Κοινότητας, φτιάχτηκαν τρία τεχνητά σύνολα δεδομένων που περιείχαν το 10%, 20% και 30% του συνόλου των βαθμολογιών, τα οποία θεωρήθηκε ότι αντιπροσωπεύουν τις αρχικές φάσεις του συστήματος.

Στα πειράματα που έγιναν, ο HIR ανταποκρίθηκε πολύ καλά και αποδείχθηκε ότι παρά το γεγονός ότι οι άμεσες σχέσεις είχαν ελάχιστη συνεισφορά στις αρχές του συστήματος, οι σχέσεις γειτνίασης που απεικονίζονται στο μητρώο $\mathbf{D}$ διατηρούνται περισσότερο. Αυτό είχε ως αποτέλεσμα, το διάνυσμα συστάσεων που επιστρέφει ο HIR να αποδεικνύεται λιγότερο ευαίσθητο στο πρόβλημα αυτό.

Για την προσομοίωση του δεύτερου προβλήματος, έγινε τυχαία επιλογή 200 χρηστών που είχαν βαθμολογήσει 100 ή περισσότερα είδη και τυχαία διαγραφή του 96%, 94% και 92% των βαθμολογιών τους, ώστε να κατασκευαστούν τρία σύνολα δεδομένων που να αφορουν νεοεισερχόμενους χρήστες.

Από τις δοκιμές που διεξήχθησαν, για τις διάφορες τιμές του β (της μεταβλητής που καθορίζει τη συμβολή του μητρώου γειτνίασης), προέκυψε η θετική συνεισφορά του μητρώου $\mathbf{D}$ στην ποιότητα της κατάταξης των συστάσεων ακόμα και για μικρές τιμές του β, αλλά και για την περίπτωση που υπήρχε διαθέσιμο μόνο το 4% των βαθμολογιών των υπό δοκιμή χρηστών.

Το αποτέλεσμα αυτό ήταν αναμενόμενο λόγω της θεώρησης του αλγορίθμου HIR καθώς παρά το ότι δεν είναι γνωστές αρκετές προτιμήσεις των χρηστών, η εκμετάλλευση της έμμεσης πληροφορίας που παρέχεται από τη γνώση των κατηγοριών που ανήκουν τα λίγα είδη που βαθμολόγησαν, δίνει στον HIR συγκριτικό πλεονέκτημα στην επιτυχή παραγωγή καλών συστάσεων.

Για τον έλεγχο της περίπτωσης των νέων ειδών, επιλέχθηκαν τυχαία το 10%, 12,5% και 15% των ειδών που είχαν τουλάχιστον 30 βαθμολογίες και αφαιρέθηκε τυχαία το 90% αυτών για τη δημιουργία 3 συνόλων δεδομένων. Και σε αυτή την περίπτωση η επίδοση του HIR ήταν πολύ καλή.

Στο Yahoo!R2Music για τις διάφορες μετρικές κατέκτησε από την 1η ως την 3η θέση και για το MovieLens1M την 1η για όλες τις μετρικές που χρησιμοποιήθηκαν, επιδεικνύοντας σταθερότητα στις κατατάξεις που επέστρεφε και έλλειψη ευαισθησίας στην αραιότητα.

Η καλή του επίδοση οφείλεται στη θεώρησή του για τις έμμεσες σχέσεις που προκύπτουν από τις βαθμολογίες των ειδών, δείχνοντας ότι ακόμα και για ανεπαρκή αριθμό βαθμολογιών, τα νέα είδη αντιμετωπίζονται πιο δίκαια.

Στα πειράματα που έγιναν, στο MovieLens100K με τη χρήση κατάλληλων μετρικών φάνηκε ότι ο αλγόριθμος έδινε τις πιο ποιοτικές κατατάξεις από όλους τους αλγορίθμους με τους οποίους συγκρίθηκε.

2.4 Συμπέρασμα

Συμπερασματικά, ο αλγόριθμος HIR εκμεταλλεύεται αποτελεσματικά τις σχέσεις των ειδών που προκύπτουν από την κατηγοριοποίησή τους και παράγει ποιοτικές κατατάξεις. Η αραιότητα έχει μειωμένη επίδραση στο μοντέλο που προτάθηκε. Επιπλέον, το μοντέλο αυτό μπορεί να υπολογιστεί αποδοτικά τόσο λόγω του ότι εξαρτάται αποκλειστικά από τη διάσταση του χώρου των ειδών όσο και λόγω των μαθηματικών ιδιοτήτων που διαθέτει. Από την πειραματική αξιολόγηση του αλγορίθμου προέκυψε η καλή συμπεριφορά του απέναντι στο πρόβλημα της κρύας εκκίνησης.

Το μοντέλο αυτό είναι επεκτάσιμο και μπορεί να εφαρμοστεί σε περαιτέρω κατηγοριοποίηση του χώρου των ειδών. Οι διάφορες υποκατηγορίες μπορουν να αναπαρασταθούν με την εισαγωγή νέων χαμηλής τάξης μητρώων $\mathbf{D}_1$, $\mathbf{D}_2, \ldots$ (και αντίστοιχων μεταβλητών β_1, $\beta_2, \ldots$) που θα απεικονίζουν με περισσότερη λεπτομέρεια την ιεραρχία του χώρου. Η εισαγωγή τέτοιων μητρώων δεν επηρεάζει τη διάσταση του μοντέλου.

Κεφάλαιο 3

Το LensKit

3.1 Εισαγωγή

Το LensKit [6] αποτελεί ένα ανοιχτού κώδικα πακέτο λογισμικού για την πραγματοποίηση μελέτης και επαληθεύσιμης έρευνας πάνω σε συστήματα παραγωγής συστάσεων. Αποτελεί μία πλατφόρμα για την ανάπτυξη αλγορίθμων παραγωγής συστάσεων, μέτρησης της επίδοσής τους πάνω σε διαφορετικά σύνολα δεδομένων και σύγκρισης νέων ιδεών με τις τρέχουσες βέλτιστες πρακτικές. [5][1] Υλοποιήθηκε από τον Michael D. Ekstrand και αναπτύχθηκε από ερευνητές στο Texas State University και στο πανεπιστήμιο της Minnesota, με συνεισφορά από προγραμματιστές από όλο τον κόσμο.[2] Αποτελείται από περισσότερες από 49K γραμμές κώδικα με συνεισφορά από 28 προγραμματιστές. Είναι γραμμένο κυρίως σε Java (92%) και ένα σημαντικό μέρος του είναι σε Groovy (7%).[3] Ο πηγαίος κώδικας του LensKit είναι δημοσιευμένος στο GitHub.[4] Η φιλοσοφία που ακολουθήθηκε για την ανάπτυξη του βασίζεται αρκετά στο [1].

Για το σκοπό που αναπτύχθηκε προσφέρει:

- Διεπαφές προγραμματισμού εφαρμογών (APIs) για την παραγωγή συστάσεων και προβλέψεων. Επιτρέπει στους προγραμματιστές να χρησιμοποιήσουν τους υλοποιημένους αλγόριθμους ως «μαύρο κουτί».
- Υλοποίηση βασικών αλγορίθμων για παραγωγή συστάσεων και πρόβλεψη βαθμολογιών.
- Εργαλειοθήκη αξιολόγησης απόδοσης σε κοινά σύνολα δεδομένων με χρήση ποικιλίας μετρικών.

[1] Κύρια πηγή του παρόντος κεφαλαίου ειναι το [5].
[2] http://lenskit.org/
[3] Στατιστικά από BlackDuck | Open Hub https://www.openhub.net/p/lenskit
[4] https://github.com/lenskit/lenskit

- Κώδικα υποστήριξης για την ανάπτυξη νέων αλγορίθμων, μεθόδων αξιολόγησης και άλλων επεκτάσεων.

Ένας παραγωγός συστάσεων στο LensKit αποτελείται από ένα σύνολο διεπαφών που παρέχουν παραγωγή συστάσεων και πρόβλεψη βαθμολογιών. Οι διεπαφές αυτές συνδέονται με μία πηγή δεδομένων χρησιμοποιώντας έναν ή περισσότερους αλγορίθμους παραγωγής συστάσεων. Η σύγκριση διαφορετικών αλγορίθμων γίνεται με τη χρήση script (σεναρίου) αφού δηλωθούν οι πηγές δεδομένων, οι αλγόριθμοι και οι μετρικές βάσει των οποίων συγκρίνονται.

3.2 Σχεδιασμός του LensKit

Ο σχεδιασμός του LensKit έγινε με στόχο τόσο την άναπτυξη και την έρευνα πάνω σε συστήματα παραγωγής συστάσεων όσο και τη χρήση του σε εκπαιδευτικούς σκοπούς. Το πανεπιστήμιο της Minnesota χρησιμοποιεί το LensKit τόσο σε μεταπτυχιακό του μάθημα όσο και σε MOOC πάνω σε συστήματα παραγωγής συστάσεων που προσφέρει μέσω της πλατφόρμας Coursera. Το LensKit αναπτύχθηκε με τρόπο που να υποστηρίζει τη διεξαγωγή πειραμάτων.[16]

Βασική αρχή στο LensKit είναι η υλοποίηση των αλγορίθμων ως ένα σύνολο σχεδόν ανεξάρτητων τμημάτων. Ένας τυπικός αλγόριθμος παραγωγής συστάσεων χρειάζεται τουλάχιστον δώδεκα διακριτά τμήματα (components) που επικοινωνούν μεταξύ τους μέσω καλώς ορισμένων διεπαφών. Αυτή η πρακτική διευκολύνει τη συντήρηση και τον έλεγχο ορθότητας καθενός από τα συστατικά της υλοποίησης.

Βασικός στόχος είναι η διασφάλιση της ορθότητας και στη συνέχεια της αποτελεσματικότητας. Στο LensKit υπάρχουν πολλά αμετάβλητα αντικείμενα (immutable objects), ώστε να διασφαλίζεται ότι ένα τμήμα δε θα επηρεάζει αρνητικά τη λειτουργικότητα ενός άλλου.

Το LensKit ακολουθεί το μοτίβο στρατηγικής (Strategy Pattern). Στο μοτίβο αυτό κατά την ανάπτυξη μιας υλοποίησης δε χρησιμοποιείται η κληρονομικότητα. Αντί για υποκλάσεις που θα υλοποιούν τους διαφορετικούς τρόπους που μία κλάση θα επιτελεί κάποια λειτουργία της, χρησιμοποιείται η χρήση ξεχωριστών τμημάτων που ορίζονται από διεπαφές.[10] Η στρατηγική αυτή έχει ως άμεσο αποτέλεσμα οι επιμέρους αλλαγές στα τμήματα που υλοποιούν τις διεπαφές να μην απαιτούν αλλαγές στις κλάσεις που τα χρησιμοποιούν. Επιπρόσθετα, επιτρέπει την επανάχρηση των τμημάτων αυτών από άλλους αλγορίθμους μειώνοντας τον απαιτούμενο κώδικα για την υλοποίηση ενός αλγορίθμου, αλλά και βοηθώντας τον ερευνητή να επικεντρωθεί μόνο στην αναγκαία υλοποίηση που χρειάζεται για να ελέγξει την υπόθεσή του.

Παρά το ότι οι υλοποιημένοι αλγόριθμοι στο LensKit είναι εξαιρετικά διαμορφώσιμοι, δεν απαιτείται από το χρήστη να εμβαθύνει σε κάθε ξεχωριστό τμήμα, ώστε να τους χρησιμοποιήσει, καθώς όπου είναι δυνατό έχουν οριστεί προεπιλογές.

Τέλος, κατά το σχεδιασμό του έχει γίνει προσπάθεια ελαχιστοποίησης των υποθέσεων που αφορούν στο είδος των δεδομένων που θα θελήσουν να χρησιμοποιήσουν οι χρήστες. Παρόμοια προσπάθεια γίνεται και όσον αφορά την υλοποίηση των ίδιων των αλγορίθμων όσο και του αξιολογητή.

3.3 Οργάνωση του κώδικα - Ενότητες

Ο κώδικας οργανώνεται σε δέκα ενότητες (modules):

1. API: Περιλαμβάνει τις διεπαφές για υψηλού επιπεδου εργασίες στην παραγωγή συστάσεων. Είναι ανεξάρτητο από τις υπόλοιπες ενότητες εκτός αυτής των δομών δεδομένων.

2. Δομές Δεδομένων (Data structures): Περιλαμβάνει τις βασικές δομές δεδομένων του LensKit.

3. Πυρήνας (Core): Περιέχει το κύριο μέρος του LensKit, εκτός του αξιολογητή και των υλοποιήσεων των αλγορίθμων. Παρέχει υποστήριξη για το χειρισμό δεδομένων και τη διαμόρφωση των αλγορίθμων.

4. Αξιολογητής (Evaluator): Περιέχει υποστήριξη για την αξιολόγηση και τη σύγκριση αλγορίθμων με χρήση γνωστών μετρικών.

5. Παραγωγοί Προβλέψεων (Predictors): Εξειδικευμένη υποστήριξη για την πρόβλεψη βαθμολογιών.

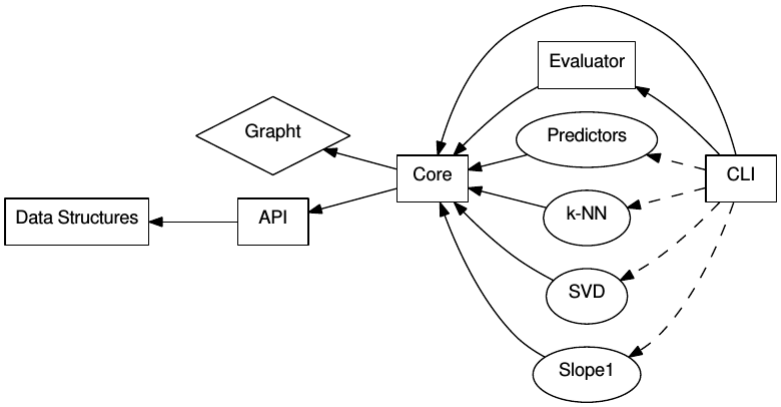
6. k-NN: Συνεργατική Διήθηση πλησιέστερου γείτονα (είδους-είδους και χρήστη-χρήστη).

7. SVD: Συνεργατική Διήθηση μέσω παραγοντοποίησης μητρώων.

8. Slope1: Υλοποίηση του αλγορίθμου Slope One.

9. Grapht: Τεχνικά δεν αποτελεί μέρος του. Είναι εργαλειοθήκη της Java που τη χρησιμοποιεί για το χειρισμό της εισαγωγής εξαρτήσεων (dependency injection).

10. CLI: Διεπαφή γραμμής εντολών για εκτέλεση αλγορίθμων, χειρισμό αρχείων δεδομένων και επιθεώρηση διαμορφώσεων αλγορίθμων.



Σχήμα 3.1: Διάγραμμα των ενοτήτων του LensKit και των μεταξύ τους σχέσεων [5]

3.3.1 Διεπαφή Παραγωγών Συστάσεων

Αποτελεί τη διεπαφή με την οποία το δημόσιο API είναι προσβάσιμο. Δεν παρέχει υλοποίηση κανενός παραγωγού συστάσεων. Χρησιμοποιείται για την επικοινωνία με τις διεπαφές των διαφόρων τμημάτων που υλοποιούν έναν παραγωγό συστάσεων και για την παροχή πρόσβασης σε εφαρμογές προς αυτά. Είναι διαχωρισμένο από τις υλοποιήσεις των παραγωγών συστάσεων και ανήκει στο δημόσιο API.

Κεντρικό τμήμα ενός παραγωγού συστάσεων είναι η υλοποίηση της διεπαφής του βαθμολογητή ειδών (Item Scorer). Ο βαθμολογητής ειδών αποτελεί γενίκευση της παραγωγής προβλέψεων και υπολογίζει προσωποποιημένες ως προς το χρήστη βαθμολογίες (ratings). Υπολογίζει τις βαθμολογίες των ειδών (scores) προβλέποντας τις βαθμολογίες των χρηστών για αυτά. Αυτό αποτελεί μια γενίκευση που επιτρέπει τη βαθμολογία ειδών ακόμα και από τις εκδηλώσεις προτίμησης των χρηστών. Ο βαθμολογητής ειδών χρησιμοποιείται έμμεσα ενώ άμεσα χρήσιμοποιούνται οι παραγωγοι συστάσεων ειδών και οι παραγωγοί προβλέψεων βαθμολογιών. Και οι τρεις έχουν την ίδια διεπαφή και αυτό που τους διαφοροποιεί είναι ότι οι τελευταίοι αναλαμβάνουν να διεκπεραιώσουν δευτερεύουσες εργασίες που απαιτούνται για την παραγωγή συστάσεων (όπως την αντιστοίχιση των βαθμολογιών σε κάποιο εύρος), ώστε να κρατείται «καθαρός» ο κώδικας του βαθμολογητή και να αποφεύγεται η επικάλυψη κώδικα.

3.4 Μοντέλο Δεδομένων

Το μοντέλο δεδομένων του LensKit στοχεύει στην αναπαράσταση και στην πρόσβαση στα δεδομένα που απαιτεί ένας παραγωγός συστάσεων. Για το σκοπό αυτό χρησιμοποιεί τις έννοιες χρήστες (users), είδη (items) και γεγονότα (events). Το μοντέλο αυτό είναι αρκετά ευέλικτο, ώστε να υποστηρίζει και άμεσες

βαθμολογίες και έμμεση εκδήλωση προτίμησης.

Οι χρήστες και τα είδη αναπαριστώνται με αριθμητικά αναγνωριστικά (numerical identifiers (Java longs).

Ως γεγονός ορίζεται η αλληλεπίδραση του χρήστη με κάποιο είδος. Για κάθε τύπο γεγονότος, ορίζεται διαφορετική διεπαφή (επεκτάση του βασικού τύπου). Μέχρι στιγμής έχουν υλοποιηθεί τρεις τύποι γεγονότων: η Βαθμολόγηση (Rating)[5], η Προτίμηση (Like)[6] και η Μαζική Προτίμηση (LikeBatch)[7]. Η Προτίμηση χρησιμοποιείται ως έκφραση απλής εκδήλωσης προτίμησης. Η Μαζική Προτίμηση αφορά τις Προτιμήσεις πολλών χρηστών.

Η επικοινωνία των διαφόρων τμημάτων του παραγωγού συστάσεων με τα δεδομένα γίνεται μεσω αντικειμένων για πρόσβαση σε δεδομένα (Data Access Objects (DAOs)). Υπάρχει ευελιξία όσον αφορά τον τρόπο που μπορεί να είναι αποθηκευμένα τα δεδομένα καθώς μπορούν να υλοποιηθούν DAOs ανάλογα με τις ανάγκες του παραγωγού συστάσεων. Το LensKit παρέχει DAOs για χειρισμό αρχείων κειμένου και βάσεων δεδομένων. Έχουν υλοποιηθεί βασικές μέθοδοι, οι οποίες επιστρέφουν βασικές δομές δεδομένων της Java και μέθοδοι συνεχούς ροής (streaming) μέθοδοι, οι οποίες επιστρέφουν κέρσορες (cursors). Οι μέθοδοι συνεχούς ροής προσφέρουν αποδοτική διαχείριση μνήμης καθώς επιτρέπουν την επεξεργασία ενός αντικειμένου κάθε φορά χωρίς να φορτώνονται όλα τα απαιτούμενα αντικείμενα στη μνήμη. Υπάρχει η δυνατότητα επέκτασης των διεπαφών για αξιοποίηση μεταδεδομένων των χρηστών ή των αντικειμένων. Για το LensKit τα DAOs είναι τμήματα, όπως αυτά που επιτελούν τις λειτουργίες ενός παραγωγού συστάσεων.

3.4.1 Δομές Δεδομένων

Για την επεξεργασία των δεδομένων είναι συχνά αναγκαία η χρήση διανυσμάτων (vectors). Για το σκοπό αυτό το LensKit προσφέρει τα Sparse Vectors (Αραιά Διανύσματα). Αποτελούν αντιστοίχιση (map) από long σε double και είναι αποδοτικά σε πράξεις γραμμικής άλγεβρας. Λειτουργούν ως παράλληλοι πίνακες αναγνωριστικών (IDs) χρηστών ή ειδών και τιμών και ταξινομούνται βάσει ID. Ο ορισμός του χώρου κλειδιών τους (key domain) γίνεται κατά τη στιγμή της δημιουργίας τους και αυτό έχει ως αποτέλεσμα αποδοτική διαχείριση της μνήμης. Τα κλειδιά για τα οποία έχει οριστεί τιμή αποτελούν το σύνολο κλειδιών (key set). Υπάρχει η αφηρημένη κλάση SparseVector , η οποία υποστηρίζει μεθοδους μόνο ανάγνωσης. Η κλάση ImmutableSparseVector, διασφαλίζει τη μη πραγματοποίηση αλλαγών στο διάνυσμα, σε αντίθεση με τη MutableSparseVector. Αν ο χώρος κλειδιών δεν είναι αρχικά γνωστός γίνεται χρήση πινάκων κατακερματισμού (hash map) και στη συνέχεια μετατροπή τους σε διανύσματα.

[5] http://lenskit.org/apidocs/org/grouplens/lenskit/data/event/Rating.html
[6] http://lenskit.org/apidocs/org/grouplens/lenskit/data/event/Like.html
[7] http://lenskit.org/apidocs/org/grouplens/lenskit/data/event/LikeBatch.html

Το LensKit χρησιμοποιεί τη βιβλιοθήκη fastutil[8]και το Google Guava[9]. Η fastutil παρέχει συλλογές (collections), οι οποίες είναι συμβατές με τη διεπαφή συλλογών της Java (Java Collection API) και επιτρέπει στο LensKit τη χρήση λιστών, συνόλων και αντιστοιχίσεων. Επιπλέον παρέχει γρήγορη επανάληψη (fast iteration), η οποία μειώνει τις απαιτήσεις δέσμευσης μνήμης.

Υπάρχει η πρόθεση αντικατάστασης των Sparse Vectors[10] είτε μέσω των δομών που προσφέρει το fastutil είτε μέσω αυτών του HPPC[11]. Επίσης, υπάρχει η σκέψη δημιουργίας νέων τύπων για τα αποτελέσματα που να υποστηρίζουν την επιστροφή και το χειρισμό περισσότερων πληροφοριών εκτός των συστάσεων, όπως του βαθμού εμπιστοσύνης σε αυτές.[12] Στόχος είναι η διατήρηση της αποδοτικότητας, αλλά και η μείωση της πολυπλοκότητας για τους προγραμματιστές.[13]

3.5 Modular αλγόριθμοι

Βασική αρχή σε όλες τις υλοποιήσεις του είναι η τμηματοποίηση των αλγορίθμων, ώστε κάθε τμήμα να εκτελεί μία και μόνο λειτουργία με στόχο την επαναχρησιμοποίηση, την ευκολότερη παραμετροποίηση και την ανάπτυξη κώδικα. Το LensKit παρέχει τις απαραίτητες υποδομές, ώστε να διευκολύνει την υλοποίηση αλγορίθμων με αυτή τη στρατηγική.

Το LensKit χρησιμοποιεί το μοτίβο στρατηγικής [10], το οποίο επιτρέπει να υπάρχουν ξεχωριστά τμήματα που επιτελούν λειτουργίες, όπως η κανονικοποίηση δεδομένων τα οποία μπορούν να χρησιμοποιηθούν από τους υλοποιημένους αλγορίθμους.

Στο LensKit γίνεται εκτεταμένη χρήση κατασκευαστών (builders). Ακολουθείται η μέθοδος του διαχωρισμού των τμημάτων που περιέχουν τα δεδομένα (data containers) (και συνοδεύονται από διεπαφές για την πρόσβαση σε αυτά) και των κατασκευαστών που κάνουν υπολογισμούς για την παραγωγή (φαινομενικά αμετάβλητων) αντικειμένων. Έτσι μπορεί εύκολα να αλλάξει η στρατηγική πραγματοποίησης των απαιτούμενων υπολογισμών χωρίς να αλλάξει η υλοποίηση του υπόλοιπου αλγορίθμου.

Επιπλέον χρησιμοποιούν τη τεχνική Builder and Facade [10] (Κατασκευή και Εκπροσώπηση), η οποία περιλαμβάνει κατασκευαστές και μία διεπαφή που λειτουργεί ως εκπρόσωπος (διεπαφή) άλλων διεπαφών.

Για την επίτευξη των παραπάνω το LensKit χρησιμοποιεί τη μέθοδο της εισαγωγής εξαρτήσεων [18], δηλαδή ένα τμήμα κώδικα όταν καλείται πρέπει να του παρέχονται και τα αντικείμενα από τα οποία εξαρτάται αντί αυτά να ενεργοποιούνται από το ίδιο το τμήμα. Με την τεχνική αυτή είναι δυνατό να αλλάξει η υλο-

[8] http://fastutil.di.unimi.it/
[9] https://github.com/google/guava
[10] https://github.com/lenskit/lenskit/wiki/ReplacingSparseVectors
[11] labs.carrotsearch.com/hppc.html
[12] https://github.com/lenskit/lenskit/wiki/DetailedResults
[13] http://mailman.cs.umn.edu/archives/lenskit/2015q1/000555.html

ποίηση κάποιας εξάρτησης και να τροποποιηθεί η συμπεριφορά του εξαρτώμενου τμήματος χωρίς αυτό να επαναπρογραμματιστεί. Για το σκοπό αυτό το LensKit χρησιμοποιεί την εργαλειοθήκη Grapht.

Τα τμήματα χωρίζονται σε αυτά που είναι προκατασκευασμένα από τα δεδομένα (pre-built) και μπορούν να χρησιμοποιηθούν από διαφορετικές κλήσεις και αυτά που χρειάζονται άμεση πρόσβαση στα δεδομένα.

3.5.1 Βασικές Υλοποιήσεις Τμημάτων

Λόγω της δομής του LensKit είναι δυνατόν να υλοποιηθεί κάποιος αλγόριθμος παραγωγής συστάσεων διαμορφώνοντας κάποιον ήδη υλοποιημένο item scorer ή υλοποιώντας μόνο το κομμάτι του αλγορίθμου που αφορά τον item scorer. Η προεπιλεγμένη υλοποίηση του item scorer είναι ο TopNItemRecommender[14],ο οποίος χρησιμοποιεί τον item scorer για να βαθμολογήσει τα είδη και επιστρέφει τα N καλύτερα (με προεπιλογή αποκλεισμού των ήδη βαθμολογημένων από τον χρήστη ειδών).

3.5.2 Παραγωγοί Περιλήψης Ιστορικού και Κανονικοποιητές

Ένας παραγωγός περιλήψης ιστορικού παράγει από το ιστορικό γεγονότων του χρήστη ένα αραιό διάνυσμα προτιμήσεων του οποίου τα κλειδιά είναι τα είδη και οι τιμές είναι κάποιες πραγματικές τιμές που αντιπροσωπεύουν τις προτιμήσεις του χρήστη. Πολλά συστήματα ενώ δε χρησιμοποιούν άμεσες βαθμολογίες, παράγουν ένα διάνυσμα που σχετίζεται με τις προτιμήσεις του χρήστη. Επίσης αυτό το διάνυσμα συχνά κανονικοποιείται ως προς το μέσο όρο των βαθμολογιών.

Οι κανονικοποιητές εφαρμόζονται πάνω σε διανύσματα αναφοράς και διανύσματα στόχους. Το διάνυσμα αναφοράς χρησιμοποιείται για τον υπολογισμό της βάσης της κανονικοποίησης και το διάνυσμα-στόχος είναι αυτό το οποίο τροποποιείται. Για λόγους αντιστρεψιμότητας υπάρχει η δυνατότητα δημιουργίας ενός μετασχηματισμού από ένα διάνυσμα αναφοράς. Ο μετασχηματισμός μετά μπορεί να εφαρμοστεί σε οποιοδήποτε διάνυσμα. Υποστηρίζονται και στοχευμένοι ως προς τον χρήστη ή το αντικείμενο κανονικοποιητές. Οι κανονικοποιητές αυτοί μπορούν να κάνουν χρήση επιπλέον πληροφορίας. Συχνά εξαρτώνται από DAOs για την πρόσβαση σε δεδομένα ή από κάποιο άλλο τμήμα το οποίο μπορεί να επωφεληθεί απο τη γνώση για το ποιο διάνυσμα κανονικοποιείται.

3.5.3 Βαθμολογητές Βάσης

Υπολογίζουν βαθμολογίες για τα είδη με χρήση απλών μέσων όρων. Χρησιμοποιούνται τόσο σε περίπτωση αποτυχίας του βασικού βαθμολογητή ειδών όσο και για κανονικοποίηση δεδομένων. Βασικοί αλγόριθμοι χρησιμοποιούν κανονικοποιημένα δεδομένα αντί για τις πραγματικές βαθμολογίες. Οι βαθμολογητές

[14] http://lenskit.org/apidocs/org/grouplens/lenskit/basic/TopNItemRecommender.html

βάσης υλοποιούν τη διεπαφή του βαθμολογητή ειδών και οποιοσδήποτε βαθμολογητής ειδών μπορεί να χρησιμοποιηθεί ως βαθμολογητής βάσης. Υποστηρίζουν τη χρήση μιας παραμέτρου απόσβεσης (damping parameter), ώστε αντικείμενα με λίγες βαθμολογίες να μην παίρνουν αδικαιολόγητα μεγάλες τιμές. Αν ο χρήστης δεν έχει βαθμολογήσει αρκετά είδη, μπορούν να ανατεθούν οι τιμές του βαθμολογητή βάσης.

3.5.4 Διαμόρφωση Αλγορίθμων

Η διαμόρφωση των αλγορίθμων γίνεται μέσω του Grapht API με τη χρήση μιας ενσωματωμένης γλώσσας σεναρίων σε Groovy, η οποία έχει πιο «ελαφριά» σύνταξη σε σχέση με τη Java. Δίνει τη δυνατότητα οι ορισμοί των αλγορίθμων να αντιμετωπίζονται ως αρχεία διαμόρφωσης (configuration files) και να μην είναι ενσωματωμένοι στον κώδικα των εφαρμογών. Η γραμμή εντολών του LensKit μπορεί να χειριστεί αυτά τα σενάρια.

3.5.5 Αξιολόγηση Αλγορίθμων και Σύνολα Δεδομένων

Υποστηρίζεται offline, προσανατολισμένη στα δεδομένα αξιολόγηση της απόδοσης μέσω εκπαίδευσης και ελέγχου (train/test) με διαχωρισμό και επαλήθευση (cross-validation). Υποστηρίζεται επεξεργασία των συνόλων δεδομένων και αξιολόγηση αλγορίθμων πάνω σε πολλαπλά σύνολα δεδομένων και μέτρηση της απόδοσής τους. Στο LensKit 3 θα γίνεται μέσω της διεπαφής γραμμής εντολών (CLI) και του Gradle[15].[16] Ο αξιολογητής χειρίζεται δεδομένα κυρίως σε μορφή delimited (οριοθετημένων) αρχείων κειμένου και δυαδικών αρχείων βαθμολογιών. Οι βασικές εργασίες χειρισμού δεδομένων είναι:

- crossfold: Διαχωρισμός της πηγής δεδομένων σε N τμήματα για cross-validation.
- pack: Μετατροπή ενός συνόλου δεδομένων σε δυαδικό αρχείο για αποδοτική πρόσβαση.
- subsample: Δημιουργία ενός μικρότερου συνόλου δεδομένων με τυχαία επιλογή από ένα μεγαλύτερο βάσει κριτηρίου.[17]

3.5.6 Αξιολόγηση Αλγορίθμων και Μετρικές Απόδοσης

Ο αξιολογητής δέχεται ένα σύνολο αλγορίθμων και ζεύγη συνόλων δεδομένων εκπαίδευσης και ελέγχου και αξιολογεί την ακρίβεια κάθε αλγορίθμου πάνω σε

[15] http://gradle.org/
[16] http://mailman.cs.umn.edu/archives/lenskit/2015q1/000560.html
[17] Θα καταργηθεί στο LensKit 3.

κάθε ζεύγος. Οι διαθέσιμες μετρικές είναι δύο ειδών[18]: μετρικές πρόβλεψης[19] και TopN μετρικές[20]. Οι μετρικές πρόβλεψης περιλαμβάνουν:

- MAE: Μέσο απόλυτο σφάλμα της ακρίβειας των προβλεφθέντων δεδομένων βαθμολόγησης.
- RMSE: Ομοία με μέσο τετραγωνικό σφάλμα.
- Coverage (κάλυψη): Μετράει τον αριθμό των παρεχόμενων προβλέψεων βαθμολογιών και υπολογίζει την κάλυψη σε σχέση με το σύνολο των ζητούμενων.
- nDCG: Αξιολογεί τις προβλέψεις του παραγωγού συστάσεων μέσω κανονικοποιημένου μειούμενου συσσωρευτικού οφέλους. Τα είδη ταξινομούνται βάσει της προβλεφθείσας προτίμησης και η μετρική υπολογίζεται χρησιμοποιώντας τις πραγματικές βαθμολογίες του χρήστη ως συνάρτηση ωφέλειας για κάθε είδος. Με αυτό τον τρόπο δεν «τιμωρεί» τον παραγωγό συστάσεων όταν προτείνει είδη τα οποία θα ήταν καλά για το χρήστη, αλλά για τα οποία δεν είχε δεδομένα.

Οι TopN μετρικές περιλαμβάνουν:

- Entropy (Εντροπία): Εξετάζει το εύρος των προτεινόμενων ειδών σε σχέση με το σύνολο των ειδών για όλους τους χρήστες. Όσο μικρότερη είναι η τιμή που επιστρέφει η μετρική τόσο λιγότερα είναι τα είδη από τα οποία προτείνει ο παραγωγός. Στη χειρότερη περίπτωση τα είδη που προτείνονται είναι τα ίδια για όλους τους χρήστες.
- Length (Μήκος): Υπολογίζει τον αριθμό των συστάσεων.
- MAP: Υπολογίζει τη μέση ακρίβεια για τους μέσους όρους ακρίβειας κάθε λίστας που επιστρέφεται.
- MRR: Υπολογίζει τη μέση αμοιβαία κατάταξη.
- nDCG: Όμοια με την αντίστοιχη μετρική πρόβλεψης.
- Popularity (Δημοφιλία): Μετράει πόσο δημοφιλή είναι τα είδη που επιστρέφονται από τον παραγωγό συστάσεων.
- Precision Recall: Ακρίβεια και ανάκληση σε σταθερού μήκους σύνολα ειδών υποψήφιων προς σύσταση και επιθυμητών ειδών. Αν στο δεύτερο σύνολο τοποθετηθούν μη επιθυμητά είδη, η μετρική μπορεί να ελέγξει αν ένας παραγωγός κάνει κακές συστάσεις.

[18] http://lenskit.org/documentation/evaluator/upgrading/
[19] http://lenskit.org/master/apidocs/org/lenskit/eval/traintest/predict/PredictMetric.html
[20] http://lenskit.org/master/apidocs/org/lenskit/eval/traintest/recommend/TopNMetric.html

3.5.7 Εισαγωγή Εξαρτήσεων

Πριν την αρχικοποίηση ενός αντικειμένου, όλες οι εξαρτήσεις πρέπει να είναι διαθέσιμες. Για την επίτευξη αυτού έχουν αναπτυχθεί εργαλειοθήκες, οι dependency injectors (εισαγωγείς εξαρτήσεων), οι οποίοι κάνουν αυτόματα την απαραίτητη διαδικασία. Το Grapht ακολουθεί πολιτική ευαίσθητη ως προς το περιεχόμενο, επιτρέποντας στα αντικείμενα να διαμορφώνονται βάσει του πού χρησιμοποιούνται. Διαχωρίζει την επίλυση εξάρτησης από την δημιουργία των στιγμιοτύπων των αντικειμένων και εκθέτει το γράφημα των επιλυμένων εξαρτήσεων των αντικειμένων ως ένα αντικείμενο που μπορεί να αναλυθεί και να διαχειρισθεί. Πολλά από τα χαρακτηριστικά του είναι υλοποιημένα με όρους μετασχηματισμού γραφημάτων.

Εισαγωγή εξάρτησης είναι σχεδιασμός που προκύπτει από την εφαρμογή αντιστροφής ελέγχου (Inversion of Control) στο πρόβλημα της δημιουργίας στιγμιοτύπων αντικειμένων που εξαρτώνται από άλλα. Μέσω αυτού αν ένα αντικείμενο A εξαρτάται από ένα αντικείμενο B, μπορεί να ζητήσει να του παρασχεθεί το B μέσω ενός ορίσματος στο δημιουργό του. Έτσι, τα τμήματα κώδικα δε γνωρίζουν για την υλοποίηση των εξαρτήσεών τους. Τα τμήματα κώδικα μπορούν να επαναδιαμορφωθούν με αλλαγή των υλοποιήσεων των εξαρτήσεών τους, χωρίς να αλλάξουν τα ίδια. Διευκολύνεται ο έλεγχος λειτουργίας τους και οι εξαρτήσεις κάθε τμήματος δηλώνονται με άμεσο τρόπο.

Κεφάλαιο 4

Ανάλυση της Υλοποίησης

Στο κεφάλαιο αυτό παρουσιάζεται η υλοποίηση του αλγορίθμου HIR. Η υλοποίηση πραγματοποιήθηκε και ελέγχθηκε σε κλώνο του master branch του LensKit, όπως αυτό υπάρχει στο GitHub. Στη συνέχεια, η υλοποίηση, εκτός των ελέγχων, ενσωματώθηκε στο Lenskit Hello project που παρέχει το LensKit και περιέχει Demo, μέσω του οποίου εκτελείται ο επιλεγμένος παραγωγός συστάσεων παίρνοντας ορίσματα από τη γραμμή εντολών. Το Demo αυτό τροποποιήθηκε, ώστε να διαμορφώνει και να χρησιμοποιεί την υλοποίηση του αλγορίθμου HIR.

4.1 HIR Item Scorer

Κεντρικό τμήμα κάθε παραγωγού συστάσεων είναι ο Item Scorer. Στο LensKit, υλοποιείται μέσω της μεθόδου:

```
ResultMap scoreWithDetails(long user, @Nonnull Collection<Long> items)
```

η οποία δέχεται ως όρισμα το αναγνωριστικό του χρήστη και μια συλλογή από τα είδη από τα οποία καλείται να παράξει συστάσεις.

Μέσω ενός αντικειμένου για την πρόσβαση στα γεγονότα που έχει συμμετάσχει ο χρήστης (βαθμολογίες), του UserEventDAO, παράγεται το ιστορικό του χρήστη, μια περίληψη του οποίου αποθηκεύεται σε ένα Sparse Vector. Στη συνέχεια, δημιουργείται το διάνυσμα προτίμησης του χρήστη, το οποίο αντιστοιχεί τα αναγνωριστικά του συνόλου των ειδών με τη βαθμολογία τους από το χρήστη. Τα αναγνωριστικά όσων ειδών δεν έχουν βαθμολογηθεί από τον χρήστη, αντιστοιχίζονται με μηδέν. Το διάνυσμα αυτό κανονικοποιείται, ώστε τα στοιχεία του να αθροίζουν στη μονάδα.

Έπειτα, εφαρμόζεται το κύριο μέρος του αλγορίθμου:

- Κατασκευάζεται το διάνυσμα κατάταξης του χρήστη, το οποίο αποτελεί αντίγραφο του διανύσματος προτίμησής του πολλαπλασιασμένο με 0.1.

- Στη συνέχεια προσπελαύνεται το σύνολο των μη μηδενικών τιμών του, οι οποίες αντιστοιχούν στα είδη για τα οποία υπάρχει πληροφορία για την αλληλεπίδραση του χρήστη με αυτά. Για αυτά τα είδη, καλείται:

  - η συνάρτηση που παρέχει το HIR Model και επιστρέφει το κανονικοποιημένο διάνυσμα που περιέχει την πληροφορία για τον αριθμό των κοινών βαθμολογίων του με άλλα είδη και

  - η συνάρτηση του HIR Model που επιστρέφει το κανονικοποιημένο διάνυσμα γειτνίασης του τρέχοντος είδους με τα υπόλοιπα βάσει των κατηγοριών στα οποία ανήκει.

- Τα παραπάνω διανύσματα αποθηκεύονται σε Sparse Vectors και πολλαπλασιάζονται με την παράμετρο άμεσης συσχέτισης και γειτνίασης αντίστοιχα. Ο γραμμικός συνδυασμός τους πολλαπλασιάζεται με τη βαθμολογία του είδους που εξετάζεται και στη συνέχεια προστίθεται με το διάνυσμα κατάταξης, όπως αυτό έχει τροποποιηθεί, υπολογίζοντας με αυτό τον τρόπο το τελικό διάνυσμα κατάταξης του χρήστη.

- Τέλος, το διάνυσμα αυτό προσπελαύνεται και τα στοιχεία του που δεν αντιπροσωπεύουν είδη τα οποία έχει δει ο χρήστης επιστρέφονται ως αποτέλεσμα της κλήσης του Item Scorer.

4.2 HIR Model

Στο HIR Model παρέχονται από το HIR Model Builder ως δεδομένα το μητρώο κοινών βαθμολογιών και τα μητρώα που αποτελούν την παραγοντοποίηση του μητρώου γειτνίασης.

Έχει δύο συναρτήσεις:

```
public MutableSparseVector getCoratingsVector(long item, Collection<Long> items)
public MutableSparseVector getProximityVector(long item, Collection<Long> items)
```

Η πρώτη ανακτά από το μητρώο άμεσης συσχέτισης το (κανονικοποιημένο) Real Vector που περιέχει τις κοινές του βαθμολογίες και αποθηκεύει τα περιεχόμενα του σε ένα Sparse Vector, το οποίο και επιστρέφει στον HIR Item Scorer.

Η δεύτερη ανακτά από το μητρώο που αποτελεί τον πρώτο παράγοντα του μητρώου γειτνίασης, το (κανονικοποιημένο) διάνυσμα γραμμή που αντιστοιχεί στο είδος που προσπελαύνεται σε μορφή Real Vector. Στη συνέχεια το πολλαπλασιάζει από δεξιά με το (κανονικοποιημένο κατά γραμμές) μητρώο που αποτελεί το δεύτερο παράγοντα του μητρώου γειτνίασης. Το αποτέλεσμα αποτελεί το διάνυσμα γειτνίασης, το οποίο αποθηκεύεται σε Real Vector και τα περιεχόμενά του αντιγράφονται στο Sparse Vector, το οποίο επιστρέφεται στον HIR Item Scorer.

4.3 HIR Model Builder

Το HIR Model Builder καλεί τις κλάσεις που παράγουν τα μητρώα του μοντέλου, τα οποία παρέχει στο HIR Model, το οποίο με τη σειρά του μετά από τους απαιραίτητους υπολογισμούς επιστρέφει στον HIR Item Scorer τα διανύσματα που χρειάζεται κάθε φορά. Επιπλέον, κατασκευάζει τα διανύσματα των ειδών που περιέχουν τις βαθμολογίες τους, τα οποία χρειάζονται για τον υπολογισμό των κοινών τους βαθμολογιών.

4.4 Direct Association Matrix

Κατασκευάζει ένα Real Matrix, το οποίο σε κάθε θέση περιέχει τον αριθμό των κοινών βαθμολογιών των ειδών που βρίσκονται στην αντίστοιχη γραμμή και στήλη του. Στη συνέχεια το κανονικοποιεί κατά γραμμές. Το μητρώο αποκτά τα δεδομένα του μέσω της συνάρτησης:

```
public void putItemPair(long id1, SparseVector itemVec1, long id2, SparseVector itemVec2)
```

και κανονικοποιείται και επιστρέφεται μέσω της:

```
public RealMatrix buildMatrix()
```

4.5 Row Stochastic Factor of Proximity

Η Row Stochastic Factor of Proximity αποθηκεύει σε ένα Real Matrix τα δεδομένα που αφορούν τις κατηγορίες που ανήκουν τα είδη (το μητρώο $\mathbf{A}$ του μοντέλου). Αποκτά πρόσβαση στην απαραίτητη πληροφορία μέσω Data Access Object, του MapItemGenreDAO.

Έπειτα μέσω της μεθόδου:

```
public RealMatrix RowStochastic()
```

το κανονικοποιεί κατά γραμμές και το επιστρέφει. Για το άθροισμα κατά γραμμές το LensKit δεν παρέχει συνάρτηση αθροίσματος, αλλά συνάρτηση για τον υπολογισμό της L1 νόρμας διανύσματος. Δεδομένου ότι τα στοιχεία του μητρώου είναι μη αρνητικά, αυτή η συνάρτηση υπολογίζει το άθροισμα κατά γραμμές.

4.6 Transposed Factor of Proximity

Εφαρμόζει την παραπάνω διαδικασία στο $\mathbf{A}^{\mathrm{T}}$.

4.7 Παράμετροι

Για τις παραμέτρους άμεσης συσχέτισης και γειτνίασης, υλοποιήθηκαν δύο διεπαφές, με τις οποίες επικοινωνεί ο HIR Item Scorer και λαμβάνει τις προεπιλεγμένες τους τιμές. Επιλέχθηκαν ως τέτοιες το $0,6$ για την άμεσης συσχέτισης και το $0,3$ για τη γειτνίασης, επειδή για αυτές ο HIR έχει καλύτερη απόδοση. [24]

4.8 Χειρισμός Δεδομένων

Για την πρόσβαση στα δεδομένα που αφορούν τις κατηγορίες των ειδών, δημιουργήθηκε η διεπαφή ItemGenreDAO, η οποία περιέχει τον ορισμό των μεθόδων:

```
RealVector getItemGenre(long item)
int getGenreSize()
```

οι οποίες χρησιμοποιούνται για την πρόσβαση στα διανύσματα που περιέχουν τις κατηγορίες στις οποίες ανήκουν τα είδη και των αριθμό των κατηγοριών αντίστοιχα.

Υλοποίηση της διεπαφής είναι το MapItemGenreDAO, το οποίο χρησιμοποιείται για το χειρισμό ενός CSV αρχείου, το οποίο περιέχει την απαραίτητη πληροφορία. Γίνεται η θεώρηση ότι αυτό το CSV περιέχει στην πρώτη θέση το id του είδους, στη δεύτερη το όνομά του και στην τρίτη ένα διάνυσμα, το οποίο έχει 1 αν το είδος ανήκει στην κατηγορία που αντιπροσωπεύει αυτή η θέση, αλλιώς 0. Τα 0 και 1 είναι χωρισμένα με «|».

4.9 Έλεγχος της υλοποίησης

Μέσω test αρχείων έγινε έλεγχος της λειτουργίας των:

- MapItemGenreDAO
- HIRItemRecommender
- HIRModelBuilder
- HIRItemScorer

Για τον έλεγχο του MapItemGenreDAO, θεωρήθηκε ένα μικρό σύνολο δεδομένων 6 ταινιών και 20 κατηγοριών, το οποίο είναι στη μορφή που αναφέρθηκε παραπάνω. Στη συνέχεια, ελέγχθηκε αν επιστρέφονται δεδομένα για ταινίες που δεν υπάρχουν πραγματικά στο σύνολο δεδομένων, ο σωστός αριθμός κατηγοριών, τα διανύσματα όπως έχουν οριστεί στο σύνολο δεδομένων και αν εντοπίζονται όλα τα αναγνωριστικά των ειδών του συνόλου δεδομένων.

Για τον HIRItemRecommender, ελέγχθηκε ότι διαμορφώνεται σωστά ο παραγωγός συστάσεων καλώντας όλα τα απαραίτητα για την υλοποίησή του τμήματα.

Για τον HIRModelBuilder θεωρήθηκε το ίδιο σύνολο δεδομένων ταινιών και δύο μικρά σύνολα βαθμολογιών και ελέγθηκε ότι επιστρέφονται τα σωστά διανύσματα άμεσης συσχέτισης και γειτνίασης.

Για τον HIRItemScorer, χρησιμοποιήθηκαν οι ίδιες ταινίες και ένα μικρό σύνολο δεδομένων τριών χρηστών που είχαν βαθμολογήσει τις ίδιες τρεις ταινίες με διαφορετικό τρόπο ο καθένας. Ελέχθηκε ότι επιστρέφονται οι βαθμολογίες που προβλέπονται θεωρητικά από τον αλγόριθμο με ακρίβεια 10^{-6}. Η επιστροφή των

σωστών βαθμολογιών διασφαλίζει ότι HIRItemRecommender θα επιστρέψει τη σωστή τους κατάταξη.

Για την παραγωγή των απαραίτητων τιμών ελέγχου για τον HIRModelBuilder και τον HIRItemScorer χρησιμοποιήθηκε το Octave.

Τέλος για το χειρισμό των εξαρτήσεων γράφτηκε ένα script σε Gradle.

4.10 Demo

Για την υλοποίηση του demo που χρησιμοποιεί τον παραγωγό συστάσεων του HIR, γράφτηκε ένα αρχείο διαμόρφωσης σε Groovy, το οποίο συνδέει τη διεπαφή του Item Scorer με την υλοποίηση του HIR Item Scorer και αρχικοποιεί τις παραμέτρους άμεσης συσχέτισης και γειτνίασης.

Για τις ανάγκες του Demo, αλλά και της συνολικής υλοποίησης, απαιτείται:

- Ένα σύνολο δεδομένων που παρέχει πληροφορίες για τα είδη, τις κατηγορίες που ανήκουν και τις βαθμολογίες των χρηστών για αυτά.
- Το τμήμα του συνόλου δεδομένων να δίνει πληροφορίες για τις κατηγορίες όπως απαιτείται από το Data Access Object που τα χειρίζεται.
- Τα είδη να έχουν συνεχόμενα αναγνωριστικά (0-index), αφού οι πληροφορίες για αυτά αποθηκεύονται σε μητρώα και όχι σε maps.

Για τις ανάγκες του Demo χρησιμοποιήθηκε η τελευταία έκδοση (8/2015) του ml-latest-small[1], το οποίο περιέχει 100.000 βαθμολογίες σε 9.000 ταινίες από 700 χρήστες. Το αρχείο που περιείχε τις ταινίες αντιγράφηκε, ώστε να τροποιηθεί και να περιέχει με κατάλληλο τρόπο τις πληροφορίες για τις κατηγορίες που ανήκουν. Η τροποποίηση του παραπάνω συνόλου δεδομένων βάσει των προϋποθέσεων που περιγράφηκαν παραπάνω έγινε με χρήστη Python script.

Το Demo διαβάζει τα αρχεία του συνόλου δεδομένων και διαμορφώνει τον HIR Item Scorer με τη βοήθεια και του Groovy script. Στη συνέχεια καλεί αυτή τη διαμόρφωση μέσω ενός Item Recommender και προτείνει από 10 ταινίες για τον χρήστη ή τους χρήστες των οποίων τα αναγνωριστικά λαμβάνει από τη γραμμή εντολών κατά την κλήση του.

4.11 Εργαλεία Υλοποίησης

Η υλοποίηση έγινε με χρήση της Community έκδοσης του IntelliJ IDEA 14.1.5. Η έκδοση της Java που χρησιμοποιήθηκε είναι η java-1.7.0-openjdk-amd64, της Python η 2.7.10 και του Octave η GNU Octave, 3.8.1. Το λειτουργικό του υπολογιστή που έγινε η υλοποίηση είναι Ubuntu 14.04.3 LTS. Ο υπολογιστής που χρησιμοποιήθηκε έχει επεξεργαστή Intel® Core™ i5-5200U CPU @ 2.20GHz

[1] http://grouplens.org/datasets/movielens/latest/

× 4 και 8GB RAM. Η συγγραφή της διπλωματικής έγινε σε LaTeX με τη χρήση TeXstudio.

Κεφάλαιο 5

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Ο αλγόριθμος HIR απαντάει στο πρόβλημα της έλλειψης πληροφορίας για τις προτιμήσεις των νέων χρηστών και τη δημοφιλία των νέων ειδών που εντάσσονται σε ένα σύστημα παραγωγής συστάσεων.

Η υλοποίηση του αλγορίθμου αυτού σε ένα σύστημα όπως το LensKit, δίνει τη δυνατότητα σε αυτή να αποτελέσει ένα τμήμα μιας νέας ενότητας που θα αφορά αλγορίθμους παραγωγής συστάσεων, οι οποίοι θα αντιμετωπίζουν το πρόβλημα της κρύας εκκίνησης. Τα εργαλεία που προσφέρει το LensKit παρέχουν την ευκαιρία σύγκρισης των διαφόρων αλγορίθμων που στοχεύουν στην ευφυή παραγωγή συστάσεων σε αυτό το τόσο προκλητικό ζήτημα που αντιμετωπίζουν όλοι οι αλγόριθμοι παραγωγής συστάσεων.

Η υλοποίηση τέτοιων αλγορίθμων στο LensKit, βοηθα τους ερευνητές να χρησιμοποιήσουν προχωρημένες τεχνικές προγραμματισμού, οι οποίες έχουν ως αποτέλεσμα κομψές υλοποιήσεις τμήματα των οποίων μπορούν να επαναχρησιμοποιηθούν, βοηθώντας στη γρηγορότερη εξέλιξη της έρευνας στον τομέα.

Κεφάλαιο 6

Κώδικας

Παρατίθεται το σύνολο του κώδικα μαζί με το path που βρίσκονται τα αντίστοιχα αρχεία στο master branch και στο Lenskit Hello. Για λόγους ευαναγνωσιμότητας δεν παρατίθεται η άδεια, τα import κάθε αρχείου και το πακέτο στο οποίο ανήκει.

HIRItemScorer.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/HIRItemScorer.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/HIRItemScorer.java

```
1  /**
2   * An {@link ItemScorer} that implements the HIR algorithm.
3   */
4  public class HIRItemScorer extends AbstractItemScorer {
5      protected final UserEventDAO dao;
6      protected final ItemDAO idao;
7      protected HIRModel model;
8      protected final PreferenceDomain domain;
9      protected double directAssociation;
10     protected double proximity;
11
12     @Inject
13     public HIRItemScorer(UserEventDAO dao,
14                          HIRModel model,
15                          ItemDAO idao,
16                          @Nullable PreferenceDomain dom,
17                          @DirectAssociationParameter double direct,
18                          @ProximityParameter double prox) {
19         this.dao = dao;
20         this.model = model;
21         this.idao = idao;
22         domain = dom;
23         directAssociation = direct;
24         proximity = prox;
25     }
26
27     @Nonnull
28     @Override
29     public ResultMap scoreWithDetails(long user, @Nonnull Collection<Long> items)
           {
30         UserHistory<Rating> history = dao.getEventsForUser(user, Rating.class);
31
```

```
32          if (history == null) { history = History.forUser(user); }
33
34          SparseVector historyVector =
                RatingVectorUserHistorySummarizer.makeRatingVector(history);
35          List<Result> results = new ArrayList<>();
36          MutableSparseVector preferenceVector =
                MutableSparseVector.create(idao.getItemIds(), 0);
37
38          double total = 0.0;
39          for (VectorEntry e: historyVector.fast()) {
40              final long key = e.getKey();
41              final double value = e.getValue();
42              preferenceVector.set(key, value);
43              total += value;
44          }
45
46          if (total != 0) {  preferenceVector.multiply(1/total); }
47
48          final double preferenceInResults = 1 - directAssociation - proximity;
49          MutableSparseVector rankingVector = preferenceVector.copy();
50          rankingVector.multiply(preferenceInResults);
51
52          for (VectorEntry e: preferenceVector.fast()) {
53              final double prefValue = e.getValue();
54              if (prefValue != 0) {
55                  final long prefKey = e.getKey();
56                  MutableSparseVector coratingsVector =
                        model.getCoratingsVector(prefKey, items);
57                  coratingsVector.multiply(directAssociation);
58                  MutableSparseVector proximityVector =
                        model.getProximityVector(prefKey, items);
59                  proximityVector.multiply(proximity);
60                  coratingsVector.add(proximityVector);
61                  coratingsVector.multiply(prefValue);
62                  rankingVector.add(coratingsVector);
63              }
64          }
65
66          for (VectorEntry e: rankingVector.fast()) {
67              final long key = e.getKey();
68              if (!historyVector.containsKey(key)) {
69                  results.add(Results.create(key, e.getValue()));
70              }
71          }
72
73          return Results.newResultMap(results);
74      }
75
76      public HIRModel getModel() { return model; }
77  }
```

HIRModel.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/HIRModel.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/HIRModel.java

```java
/**
 * A model for a {@link HIRItemScorer}.
 * Stores calculated proximity values and number of co-rating users for each item pair.
 */
@DefaultProvider(HIRModelBuilder.class)
@Shareable
@SuppressWarnings("deprecation")
public class HIRModel implements Serializable {
    private static final long serialVersionUID  = 1L;
    private final RealMatrix cmatrix;
    private final RealMatrix xmatrix;
    private final RealMatrix ymatrix;

    public HIRModel(RealMatrix cmatrix, RealMatrix xmatrix, RealMatrix ymatrix) {
        this.cmatrix = cmatrix;
        this.xmatrix = xmatrix;
        this.ymatrix = ymatrix;
    }
    public MutableSparseVector getCoratingsVector(long item, Collection<Long> items) {
        RealVector data = cmatrix.getRowVector((int) item);
        Map<Long, Double> forResults = new HashMap<>();
        LongIterator iter = LongIterators.asLongIterator(items.iterator());
        int i = 0;
        while (iter.hasNext()) {
            final long meti = iter.nextLong();
            forResults.put(meti, data.getEntry(i));
            i++;
        }
        return MutableSparseVector.create(forResults);
    }
    public MutableSparseVector getProximityVector(long item, Collection<Long> items) {
        RealVector data = xmatrix.getRowVector((int) item);
        RealVector resM = ymatrix.preMultiply(data);
        Map<Long, Double> forResults = new HashMap<>();
        LongIterator iter = LongIterators.asLongIterator(items.iterator());
        int i = 0;
        while (iter.hasNext()) {
            final long meti = iter.nextLong();
            forResults.put(meti, resM.getEntry(i));
            i++;
        }
        return MutableSparseVector.create(forResults);
    }
}
```

HIRModelBuilder.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/HIRModelBuilder.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/HIRModelBuilder.java

```
/**
 * Pre-computes the number of mutual rating users for every pair
 * of items and stores the results in a {@code DirectAssociationMatrix}.
 * Provides this matrix and the matrices that factorize the {
 * @code ProximityMatrix} to the model.
 * These matrices are later used by a
 * {@code HIRItemScorer}.
 */
@SuppressWarnings("deprecation")
public class HIRModelBuilder implements Provider<HIRModel> {

    private final DirectAssociationMatrix DAMatrix;
    private final RowStochasticFactorOfProximity RSMatrix;
    private final TransposedFactorOfProximity TFMatrix;
    private final ItemItemBuildContext buildContext;

    @Inject
    public HIRModelBuilder(@Transient @Nonnull ItemDAO dao,
                           @Transient @Nonnull ItemGenreDAO gDao,
                           @Transient ItemItemBuildContext context) {
        buildContext = context;
        DAMatrix = new DirectAssociationMatrix(dao);
        RSMatrix = new RowStochasticFactorOfProximity(dao, gDao);
        TFMatrix = new TransposedFactorOfProximity(dao, gDao);
    }

    /**
     * Constructs and returns a {@link HIRModel}.
     */

    @Override
    public HIRModel get() {
        LongSet items = buildContext.getItems();
        LongIterator outer = items.iterator();
        while (outer.hasNext()) {
            final long item1 = outer.nextLong();
            final SparseVector vec1 = buildContext.itemVector(item1);
            LongIterator inner = items.iterator();
            while (inner.hasNext()) {
                final long item2 = inner.nextLong();
                SparseVector vec2 = buildContext.itemVector(item2);
                DAMatrix.putItemPair(item1, vec1, item2, vec2);
            }
        }
        return new HIRModel(DAMatrix.buildMatrix(), RSMatrix.RowStochastic(),
             TFMatrix.ColumnStochastic());
    }
}
```

DirectAssociationMatrix.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/DirectAssociationMatrix.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/DirectAssociationMatrix.java

```
1  /**
2   * A matrix to store the direct inter-item relationships
3   * that derive from the number of their coratings.
4   */
5  public class DirectAssociationMatrix {
6      private RealMatrix workMatrix;
7      private int itemSize;
8      /**
9       * Creates a matrix to process rating data and generate coratings for
10      * a {@code HIRItemScorer}.
11      * @param dao       The DataAccessObject interfacing with the data for the
            model
12      */
13     public DirectAssociationMatrix(ItemDAO dao) {
14         LongSet items = dao.getItemIds();
15         itemSize = items.size();
16         workMatrix = MatrixUtils.createRealMatrix(itemSize, itemSize);
17     }
18     /**
19      * Puts the item pair into the accumulator.
20      * @param id1      The id of the first item.
21      * @param itemVec1 The rating vector of the first item.
22      * @param id2      The id of the second item.
23      * @param itemVec2 The rating vector of the second item.
24      */
25     public void putItemPair(long id1, SparseVector itemVec1, long id2,
            SparseVector itemVec2) {
26         if (id1 == id2) { workMatrix.setEntry((int) id1, (int)id2, 0); }
27         else {
28             long coratings = 0;
29             for (Pair<VectorEntry,VectorEntry> pair:
                    Vectors.fastIntersect(itemVec1, itemVec2)) {
30                 coratings++;
31             }
32             workMatrix.setEntry((int) id1, (int) id2, coratings);
33         }
34     }
35     /**
36      * @return A matrix of item corating values to be used by a {@code
            HIRItemScorer}.
37      */
38     public RealMatrix buildMatrix() {
39         for (int i=0; i<itemSize; i++) {
40             RealVector testRow = workMatrix.getRowVector(i);
41             double testSum = testRow.getL1Norm();
42             if (testSum != 0){
43                 testRow.mapDivideToSelf(testSum);
44                 workMatrix.setRowVector(i, testRow);
45             }
46         }
47         return workMatrix;
48     }
49 }
```

RowStochasticFactorOfProximity.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/RowStochasticFactorOfProximity.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/RowStochasticFactorOfProximity.java

```
1  /**
2   * A matrix to store the first factor of the matrix
3   * that contains the inter-item relationships
4   * that derive from their categorization.
5   */
6
7  public class RowStochasticFactorOfProximity {
8      private RealMatrix rowStochastic;
9      private int itemSize;
10
11     /**
12      * Creates a matrix to process genre data and generate the first factor of
            the proximity
13      * matrix needed for a {@code HIRItemScorer}.
14      *
15      * @param dao    The DataAccessObject interfacing with the item data for the
            model
16      * @param gDao   The genreDataAccessObject interfacing with the genre data
            for the model
17      *
18      */
19     public RowStochasticFactorOfProximity(ItemDAO dao, ItemGenreDAO gDao) {
20         LongSet items = dao.getItemIds();
21         int genreSize = gDao.getGenreSize();
22         itemSize = items.size();
23         double[][] data = new double[itemSize][genreSize];
24
25         rowStochastic = MatrixUtils.createRealMatrix(data);
26
27         int i = 0;
28         LongIterator iter = items.iterator();
29         while (iter.hasNext()) {
30             long item = iter.nextLong();
31             rowStochastic.setRowVector(i, gDao.getItemGenre(item));
32             i++;
33         }
34     }
35
36     /**
37      * @return A matrix containing the row stochastic values of the matrix
38      * that contains the information about the item categorization,
39      * to be used by a {@code HIRItemScorer}.
40      */
41     public RealMatrix RowStochastic() {
42         for (int i = 0; i < itemSize; i++) {
43             RealVector forIter = rowStochastic.getRowVector(i);
44
45             double sum = forIter.getL1Norm();
46             if (sum!=0) {
47                 forIter.mapDivideToSelf(sum);
48                 rowStochastic.setRowVector(i, forIter);
49             }
50         }
51         return rowStochastic;
52     }
53 }
```

TransposedFactorOfProximity.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/TransposedFactorOfProximity.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/TransposedFactorOfProximity.java

```
1  /**
2   * A matrix to store the second factor of the matrix
3   * that contains the inter-item relationships
4   * that derive from their categorization.
5   */
6
7  public class TransposedFactorOfProximity {
8      private RealMatrix transposed;
9      private int genreSize;
10
11     /**
12      * Creates a matrix to process genre data and generate the second factor of
            the proximity
13      * matrix needed for a {@code HIRItemScorer}.
14      *
15      * @param dao    The DataAccessObject interfacing with the item data for the
            model
16      * @param gDao   The genreDataAccessObject interfacing with the genre data
            for the model
17      *
18      */
19
20     public TransposedFactorOfProximity(ItemDAO dao, ItemGenreDAO gDao) {
21         LongSet items = dao.getItemIds();
22         genreSize = gDao.getGenreSize();
23         int itemSize = items.size();
24         double[][] dataTransposed = new double[genreSize][itemSize];
25         transposed = MatrixUtils.createRealMatrix(dataTransposed);
26
27         int i = 0;
28         LongIterator iter = items.iterator();
29         while (iter.hasNext()) {
30             long item = iter.nextLong();
31             transposed.setColumnVector(i, gDao.getItemGenre(item));
32             i++;
33         }
34     }
35
36     /**
37      * @return A matrix containing the row stochastic values of the matrix
38      * that contains the information about the item categorization transposed,
39      * to be used by a {@code HIRItemScorer}.
40      */
41
42     public RealMatrix ColumnStochastic() {
43
44         for (int i = 0; i < genreSize; i++) {
45             RealVector forIter = transposed.getRowVector(i);
46             double sum = forIter.getL1Norm();
47             if (sum!=0){
48                 forIter.mapDivideToSelf(sum);
49                 transposed.setRowVector(i, forIter);
50             }
51         }
52
53         return transposed;
54     }
55 }
```

DirectAssociationParameter.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/DirectAssociationParameter.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/DirectAssociationParameter.java

```
/**
 * Parameter for Direct Association Matrix.
 * It determines how much is the ranking vector returned from
 * {@code HIRItemScorer} affected by the coratings.
 */

@Documented
@DefaultDouble(0.6)
@Parameter(Double.class)
@Qualifier
@Target({ElementType.METHOD, ElementType.PARAMETER})
@Retention(RetentionPolicy.RUNTIME)
public @interface DirectAssociationParameter {
}
```

ProximityParameter.java
lenskit/lenskit-hir/src/main/java/org/grouplens/lenskit/hir/ProximityParameter.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/ProximityParameter.java

```
/**
 * Proximity Parameter.
 * It determines how much is the ranking vector returned from
 * {@code HIRItemScorer} affected by the inter-item relationships
 * that derive from the categorization of items.
 */

@Documented
@DefaultDouble(0.3)
@Parameter(Double.class)
@Qualifier
@Target({ElementType.METHOD, ElementType.PARAMETER})
@Retention(RetentionPolicy.RUNTIME)
public @interface ProximityParameter {
}
```

ItemGenreDAO.java
lenskit/lenskit-core/src/main/java/org/lenskit/data/dao/ItemGenreDAO.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/ItemGenreDAO.java

```
public interface ItemGenreDAO {
    /**
     * Get the genre for an item.
     * @param item The item ID.
     * @return A display genre for the item.
     */
    @Nullable
    RealVector getItemGenre(long item);

    /**
     * Get the number of genres in the dataset.
     * @return the number of genres in the dataset.
     */
    int getGenreSize();
}
```

MapItemGenreDAO.java
lenskit/lenskit-core/src/main/java/org/lenskit/data/dao/MapItemGenreDAO.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/MapItemGenreDAO.java

```
1  /**
2   * An item genre DAO backed by a map of item IDs to genres.
3   *
4   * @see org.lenskit.data.dao.ItemGenreDAO
5   */
6
7  public class MapItemGenreDAO implements ItemGenreDAO, ItemDAO, Serializable {
8      private static final Logger logger =
           LoggerFactory.getLogger(MapItemGenreDAO.class);
9      private static final long serialVersionUID = 1L;
10     private final Map<Long, RealVector> itemGenreMap;
11     private final LongSortedSet itemIds;
12     private static int genreSize = 0;
13
14     public MapItemGenreDAO(Map<Long, RealVector> items) {
15         itemGenreMap = ImmutableMap.copyOf(items);
16         itemIds = LongUtils.packedSet(itemGenreMap.keySet());
17     }
18
19     @Nullable
20     @Override
21     public LongSet getItemIds() {
22         return itemIds;
23     }
24
25     @Nullable
26     @Override
27     public RealVector getItemGenre(long item) {
28         return itemGenreMap.get(item);
29     }
30
31     @Override
32     public int getGenreSize() { return genreSize; }
33
34
35     /**
36      * Read an item list DAO from a file with no header rows.
37      * @param file A file of item IDs, one per line.
38      * @return The item list DAO.
39      * @throws java.io.IOException if there is an error reading the list of items.
40      */
41     public static MapItemGenreDAO fromCSVFile(File file) throws IOException {
42         return fromCSVFile(file, 0);
43     }
44
45     /**
46      * Read an item list DAO from a file.
47      * @param file A file of item IDs, one per line.
48      * @param skipLines The number of initial header to skip
49      * @return The item list DAO.
50      * @throws java.io.IOException if there is an error reading the list of items.
51      */
52     public static MapItemGenreDAO fromCSVFile(File file, int skipLines) throws
           IOException {
53         Preconditions.checkArgument(skipLines >= 0, "cannot skip negative lines");
54         LineStream stream = LineStream.openFile(file, CompressionMode.AUTO);
55         try {
56             ObjectStreams.consume(skipLines, stream);
```

```
57              ImmutableMap.Builder<Long, RealVector> genres =
                     ImmutableMap.builder();
58              StrTokenizer tok = StrTokenizer.getCSVInstance();
59              for (String line : stream) {
60                  tok.reset(line);
61                  long item = Long.parseLong(tok.next());
62                  String title = tok.nextToken();
63                  String genre = tok.nextToken();
64                  if (genre != null) {
65                      StrTokenizer gen = new StrTokenizer(genre, "|");
66                      genreSize = gen.size();
67                      double[] genValues = new double[genreSize];
68                      int i = 0;
69                      while (gen.hasNext()) {
70                          double genValue = Double.parseDouble(gen.nextToken());
71                          genValues[i] = genValue;
72                          i++;
73                      }
74                      RealVector genVec = MatrixUtils.createRealVector(genValues);
75                      genres.put(item, genVec);
76                  }
77              }
78              return new MapItemGenreDAO(genres.build());
79          } catch (NoSuchElementException ex) {
80              throw new IOException(String.format("%s:%s: not enough columns",
81                                                  file, stream.getLineNumber()),
82                                    ex);
83          } catch (NumberFormatException ex) {
84              throw new IOException(String.format("%s:%s: id not an integer",
85                                                  file, stream.getLineNumber()),
86                                    ex);
87          } finally {
88              stream.close();
89          }
90      }
91  }
```

Το παρακάτω αρχείο παρέχει κώδικα για πιθανή χρήση του από script αξιολόγησης και δεν αποτελεί μέρος της υλοποίησης.

CSVFileItemGenreDAOProvider.java

lenskit/lenskit-core/src/main/java/org/grouplens/lenskit/data/text/CSVFileItemGenreDAOProvider.java

```
1  /**
2   * Provider for {@link org.lenskit.data.dao.ItemListItemDAO}
3   * that reads a list of item IDs from a file, one per line.
4   */
5  public class CSVFileItemGenreDAOProvider implements Provider<MapItemGenreDAO> {
6     private final File itemFile;
7
8     @Inject
9     public CSVFileItemGenreDAOProvider(@ItemFile File file) {
10              itemFile = file;
11          }
12
13    @Override
14    public MapItemGenreDAO get() {
15          try {
16              return MapItemGenreDAO.fromCSVFile(itemFile);
17          } catch (IOException e) {
18              throw new DataAccessException("error reading " + itemFile, e);
19          }
20    }
21 }
```

HelloLenskit.java
lenskit-hello/src/main/java/org/grouplens/lenskit/hello/HelloLenskit.java

```java
/**
 * Demonstration app for LensKit. This application builds an item-item CF model
 * from a CSV file, then generates recommendations for a user.
 *
 * Usage: java org.grouplens.lenskit.hello.HelloLenskit ratings.csv user
 */
public class HelloLenskit implements Runnable {
    public static void main(String[] args) {
        HelloLenskit hello = new HelloLenskit(args);
        try {
            hello.run();
        } catch (RuntimeException e) {
            System.err.println(e.toString());
            e.printStackTrace(System.err);
            System.exit(1);
        }
    }

    private File inputFile = new File("data/ratings.csv");
    private File movieFile = new File("data/movies.csv");
    private File genreFile = new File("data/genres.csv");

    private List<Long> users;

    public HelloLenskit(String[] args) {
        users = new ArrayList<>(args.length);
        for (String arg: args) {
            users.add(Long.parseLong(arg));
        }
    }

    public void run() {
        EventDAO dao = TextEventDAO.create(inputFile, Formats.movieLensLatest());
        ItemNameDAO names;
        MapItemGenreDAO genres;

        try {
            names = MapItemNameDAO.fromCSVFile(movieFile, 1);
        } catch (IOException e) {
            throw new RuntimeException("cannot load names", e);
        }

        try {
            genres = MapItemGenreDAO.fromCSVFile(genreFile);
        } catch (IOException g) {
            throw new RuntimeException("cannot load names", g);
        }

        // Next: load the LensKit algorithm configuration
        LenskitConfiguration config = null;
        try {
            config = ConfigHelpers.load(new File("etc/hir.groovy"));
        } catch (IOException e) {
            throw new RuntimeException("could not load configuration", e);
        }
        // Add our data component to the configuration

        config.addComponent(dao);
        config.bind(EventDAO.class).to(dao);
```

```
60          config.bind(MapItemGenreDAO.class).to(genres);
61          config.bind(PreferenceDomain.class).to(new PreferenceDomain(0, 1));
62
63          LenskitRecommenderEngine engine = LenskitRecommenderEngine.build(config);
64
65          try (LenskitRecommender rec = engine.createRecommender()) {
66              ItemRecommender irec = rec.getItemRecommender();
67              assert irec != null;
68              for (long user : users) {
69                  // get 10 recommendation for the user
70                  List<Result> recs = irec.recommendWithDetails(user, 10, null,
                        null);
71                  System.out.format("Recommendations for user %d:\n", user);
72                  for (Result item : recs) {
73                      String name = names.getItemName(item.getId());
74                      System.out.format("\t %s \n", name);
75                  }
76              }
77          }
78      }
79  }
```

hir.groovy
lenskit-hello/etc/hir.groovy

```
1  import org.grouplens.lenskit.hello.DirectAssociationParameter
2  import org.grouplens.lenskit.hello.HIRItemScorer
3  import org.grouplens.lenskit.hello.ProximityParameter
4  import org.lenskit.api.ItemScorer
5
6  // Configuration of the item scorer.
7  bind ItemScorer to HIRItemScorer.class
8
9  set DirectAssociationParameter to 0.6
10 set ProximityParameter to 0.3
```

MapItemGenreDAOTest.java
lenskit/lenskit-core/src/test/java/org/lenskit/data/dao/MapItemGenreDAOTest.java

```
/**
 * Tests MapItemGenreDAO.
 */

public class MapItemGenreDAOTest {
    @Rule
    public TemporaryFolder folder = new TemporaryFolder();
    MapItemGenreDAO gdao;

    @Before
    public void createFile() throws IOException {
        File f = folder.newFile("genres.csv");
        PrintStream str = new PrintStream(f);
        try {
            str.println("0,\"Shawshank Redemption, The
                (1994)\",0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
            str.println("1,American History X
                (1998),0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
            str.println("2,Z (1969),0|0|0|0|0|0|0|1|0|0|0|0|1|0|0|1|0|0|0|0");
            str.println("3,\"Pan's Labyrinth (Laberinto del fauno, El)
                (2006)\",0|0|0|0|0|0|0|1|1|0|0|0|0|0|1|0|0|0|0");
            str.println("4,Seven Pounds
                (2008),0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0");
            str.println("5,Song of the Sea
                (2014),0|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0");
        } finally {
            str.close();
        }
        gdao = MapItemGenreDAO.fromCSVFile(f);
    }
    @Test
    public void testMissingItem() {
        assertThat(gdao.getItemGenre(6), nullValue());
    }
    @Test
    public void testGenreSize() {
        assertThat(gdao.getGenreSize(), equalTo(20));
    }
    @Test
    public void testGenreVector() {
        double[] testVec1 = {0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0};
        double[] testVec2 = {0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0};
        RealVector testRealVector1 = MatrixUtils.createRealVector(testVec1);
        RealVector testRealVector2 = MatrixUtils.createRealVector(testVec2);
        assertThat(gdao.getItemGenre(0), equalTo(testRealVector1));
        assertThat(gdao.getItemGenre(5), equalTo(testRealVector2));
        assertThat(testRealVector1.getDimension(), equalTo(gdao.getGenreSize()));
        assertThat(gdao.getItemGenre(0).getDimension(),
             equalTo(gdao.getGenreSize()));
        assertThat(testVec1.length, equalTo(gdao.getGenreSize()));
    }
    @Test
    public void testItemIds() {
        assertThat(gdao.getItemIds(), containsInAnyOrder(0L, 1L, 2L, 3L, 4L, 5L));
    }
}
```

HIRItemRecommenderTest.java
lenskit/lenskit-hir/src/test/java/org/lenskit/hir/HIRItemRecommenderTest.java

```
1  /**
2   * Tests if the HIR Item Recommender is created
3   * and properly configured.
4   */
5
6  public class HIRItemRecommenderTest {
7      private LenskitRecommenderEngine engine;
8
9      @Rule
10     public TemporaryFolder folder = new TemporaryFolder();
11     MapItemGenreDAO gdao;
12     ItemDAO idao;
13     EventDAO dao;
14
15     @Before
16     public void createFile() throws IOException {
17         File f = folder.newFile("genres.csv");
18         PrintStream str = new PrintStream(f);
19         try {
20             str.println("0,\"Shawshank Redemption, The
                   (1994)\",0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
21             str.println("1,American History X
                   (1998),0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
22             str.println("2,Z (1969),0|0|0|0|0|0|0|1|0|0|0|0|1|0|0|1|0|0|0");
23             str.println("3,\"Pan's Labyrinth (Laberinto del fauno, El)
                   (2006)\",0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|1|0|0|0|0");
24             str.println("4,Seven Pounds
                   (2008),0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0");
25             str.println("5,Song of the Sea
                   (2014),0|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0");
26         } finally {
27             str.close();
28         }
29         gdao = MapItemGenreDAO.fromCSVFile(f);
30     }
31
32     @SuppressWarnings("deprecation")
33     @Before
34     public void setup() throws RecommenderBuildException {
35         List<Rating> rs = new ArrayList<>();
36         rs.add(Rating.create(1, 0, 2));
37         rs.add(Rating.create(1, 1, 4));
38         rs.add(Rating.create(2, 2, 5));
39         rs.add(Rating.create(2, 3, 4));
40
41         dao = new EventCollectionDAO(rs);
42         idao = new ItemListItemDAO(LongUtils.packedSet(0, 1, 2, 3, 4, 5));
43
44
45         LenskitConfiguration config = new LenskitConfiguration();
46         config.bind(EventDAO.class).to(dao);
47         config.bind(ItemDAO.class).to(idao);
48         config.bind(MapItemGenreDAO.class).to(gdao);
49         config.bind(ItemScorer.class).to(HIRItemScorer.class);
50         config.bind(PreferenceDomain.class).to(new PreferenceDomain(0, 1));
51         config.bind(BaselineScorer.class, ItemScorer.class)
52               .to(UserMeanItemScorer.class);
53         config.bind(UserMeanBaseline.class, ItemScorer.class)
54               .to(ItemMeanRatingItemScorer.class);
```

```
55          engine = LenskitRecommenderEngine.build(config);
56      }
57
58      @SuppressWarnings("deprecation")
59      @Test
60      public void testHIRRecommenderEngineCreate() {
61          try (Recommender rec = engine.createRecommender()) {
62              assertThat(rec.getItemScorer(),
63                         instanceOf(HIRItemScorer.class));
64              RatingPredictor rp = rec.getRatingPredictor();
65              assertThat(rp, notNullValue());
66              assertThat(rp, instanceOf(SimpleRatingPredictor.class));
67              assertThat(((SimpleRatingPredictor) rp).getItemScorer(),
68                         sameInstance(rec.getItemScorer()));
69              assertThat(rec.getItemRecommender(),
70                         instanceOf(TopNItemRecommender.class));
71          }
72      }
73
74      @Test
75      public void testConfigSeparation() {
76          try (LenskitRecommender rec1 = engine.createRecommender();
77               LenskitRecommender rec2 = engine.createRecommender()) {
78
79              assertThat(rec1.getItemScorer(),
80                         not(sameInstance(rec2.getItemScorer())));
81              assertThat(rec1.get(HIRModel.class),
82                         allOf(not(nullValue()),
83                               sameInstance(rec2.get(HIRModel.class))));
84          }
85      }
86  }
```

HIRModelBuilderTest.java
lenskit/lenskit-hir/src/test/java/org/lenskit/hir/HIRModelBuilderTest.java

```
/**
 * Tests HIR Model Builder
 */
@SuppressWarnings("deprecation")
public class HIRModelBuilderTest {
    @Rule
    public TemporaryFolder folder = new TemporaryFolder();
    MapItemGenreDAO gdao;
    MapItemNameDAO idao;
    Collection<Long> items = new HashSet<>();
    List<Rating> rs1 = new ArrayList<>();
    List<Rating> rs2 = new ArrayList<>();

    @Before
    public void createFile() throws IOException {
        File f = folder.newFile("genres.csv");
        PrintStream str = new PrintStream(f);
        try {
            str.println("0,\"Shawshank Redemption, The
                (1994)\",0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0");
            str.println("1,American History X
                (1998),0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0");
            str.println("2,Z (1969),0|0|0|0|0|0|0|1|0|0|0|0|1|0|0|1|0|0");
            str.println("3,\"Pan's Labyrinth (Laberinto del fauno, El)
                (2006)\",0|0|0|0|0|0|0|1|1|0|0|0|0|0|1|0|0|0");
            str.println("4,Seven Pounds
                (2008),0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0");
            str.println("5,Song of the Sea
                (2014),0|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|0");
        } finally {
            str.close();
        }
        gdao = MapItemGenreDAO.fromCSVFile(f);
        idao = MapItemNameDAO.fromCSVFile(f);
        items.add((long)0);
        items.add((long)1);
        items.add((long)2);
        items.add((long)3);
        items.add((long)4);
        items.add((long)5);
    }

    @Before
    public void createRating1() throws IOException {
        rs1.add(Rating.create(1,0,5));
        rs1.add(Rating.create(1,2,5));
        rs1.add(Rating.create(2,0,4));
        rs1.add(Rating.create(2,2,4));
        rs1.add(Rating.create(3,1,5));
        rs1.add(Rating.create(4,1,1));
    }

    @Before
    public void createRating2() throws IOException {
        rs2.add(Rating.create(1, 0, 4));
        rs2.add(Rating.create(1, 4, 3));
        rs2.add(Rating.create(1, 5, 1));
        rs2.add(Rating.create(2, 0, 4));
        rs2.add(Rating.create(2, 4, 4));
```

```
55          rs2.add(Rating.create(2, 5, 4));
56          rs2.add(Rating.create(3, 0, 1));
57          rs2.add(Rating.create(3, 4, 1));
58          rs2.add(Rating.create(3, 5, 3));
59      }
60
61      private HIRModel getModel(List<Rating> rs) {
62          EventDAO dao = EventCollectionDAO.create(rs);
63          UserEventDAO udao = new PrefetchingUserEventDAO(dao);
64          ItemDAO idao = new ItemListItemDAO(LongUtils.packedSet(0, 1, 2, 3, 4, 5));
65          UserHistorySummarizer summarizer = new
                RatingVectorUserHistorySummarizer();
66          ItemItemBuildContextProvider contextFactory = new
                ItemItemBuildContextProvider(
67                  udao, new DefaultUserVectorNormalizer(), summarizer);
68          HIRModelBuilder provider = new HIRModelBuilder(idao, gdao,
                contextFactory.get());
69          return provider.get();
70      }
71
72      @Test
73      public void testBuild1() {
74          HIRModel model1 = getModel(rs1);
75          MutableSparseVector msv1 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
76          MutableSparseVector msv2 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
77          MutableSparseVector msv3 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
78          MutableSparseVector msv4 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
79          MutableSparseVector msv5 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
80          MutableSparseVector msv6 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
81
82          msv1.set(0, 0);
83          msv1.set(1, 0);
84          msv1.set(2, 1);
85          msv1.set(3, 0);
86          msv1.set(4, 0);
87          msv1.set(5, 0);
88
89          msv2.set(0, 0);
90          msv2.set(1, 0);
91          msv2.set(2, 0);
92          msv2.set(3, 0);
93          msv2.set(4, 0);
94          msv2.set(5, 0);
95
96          msv3.set(0, 1);
97          msv3.set(1, 0);
98          msv3.set(2, 0);
99          msv3.set(3, 0);
100         msv3.set(4, 0);
101         msv3.set(5, 0);
102
103         msv4.set(0, 0);
104         msv4.set(1, 0);
105         msv4.set(2, 0);
106         msv4.set(3, 0);
107         msv4.set(4, 0);
108         msv4.set(5, 0);
109
110         msv5.set(0, 0);
111         msv5.set(1, 0);
112         msv5.set(2, 0);
113         msv5.set(3, 0);
```

```
114         msv5.set(4, 0);
115         msv5.set(5, 0);
116 
117         msv6.set(0, 0);
118         msv6.set(1, 0);
119         msv6.set(2, 0);
120         msv6.set(3, 0);
121         msv6.set(4, 0);
122         msv6.set(5, 0);
123 
124         assertEquals(msv1, model1.getCoratingsVector(0, items));
125         assertEquals(msv2, model1.getCoratingsVector(1, items));
126         assertEquals(msv3, model1.getCoratingsVector(2, items));
127         assertEquals(msv4, model1.getCoratingsVector(3, items));
128         assertEquals(msv5, model1.getCoratingsVector(4, items));
129         assertEquals(msv6, model1.getCoratingsVector(5, items));
130     }
131 
132     @Test
133     public void testBuild2() {
134         HIRModel model2 = getModel(rs2);
135 
136         MutableSparseVector msv0 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
137         MutableSparseVector msv1 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
138         MutableSparseVector msv2 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
139         MutableSparseVector msv3 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
140         MutableSparseVector msv4 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
141         MutableSparseVector msv5 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
142         MutableSparseVector pv0 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
143         MutableSparseVector pv1 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
144         MutableSparseVector pv2 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
145         MutableSparseVector pv3 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
146         MutableSparseVector pv4 = MutableSparseVector.create(0, 1, 2, 3, 4, 5);
147 
148         msv0.set(0, 0);
149         msv0.set(1, 0);
150         msv0.set(2, 0);
151         msv0.set(3, 0);
152         msv0.set(4, 0.5);
153         msv0.set(5, 0.5);
154 
155         msv1.set(0, 0);
156         msv1.set(1, 0);
157         msv1.set(2, 0);
158         msv1.set(3, 0);
159         msv1.set(4, 0);
160         msv1.set(5, 0);
161 
162         msv2.set(0, 0);
163         msv2.set(1, 0);
164         msv2.set(2, 0);
165         msv2.set(3, 0);
166         msv2.set(4, 0);
167         msv2.set(5, 0);
168 
169         msv3.set(0, 0);
170         msv3.set(1, 0);
171         msv3.set(2, 0);
172         msv3.set(3, 0);
173         msv3.set(4, 0);
174         msv3.set(5, 0);
175 
```

```
176             msv4.set(0, 0.5);
177             msv4.set(1, 0);
178             msv4.set(2, 0);
179             msv4.set(3, 0);
180             msv4.set(4, 0);
181             msv4.set(5, 0.5);
182
183             msv5.set(0, 0.5);
184             msv5.set(1, 0);
185             msv5.set(2, 0);
186             msv5.set(3, 0);
187             msv5.set(4, 0.5);
188             msv5.set(5, 0);
189
190             pv0.set(0, 7 / 20.0);
191             pv0.set(1, 7 / 20.0);
192             pv0.set(2, 1 / 10.0);
193             pv0.set(3, 1 / 10.0);
194             pv0.set(4, 1 / 10.0);
195             pv0.set(5, 0.0);
196
197             pv1.set(0, 7 / 20.0);
198             pv1.set(1, 7 / 20.0);
199             pv1.set(2, 1 / 10.0);
200             pv1.set(3, 1 / 10.0);
201             pv1.set(4, 1 / 10.0);
202             pv1.set(5, 0);
203
204             pv2.set(0, 1 / 15.0);
205             pv2.set(1, 1 / 15.0);
206             pv2.set(2, 17 / 30.0);
207             pv2.set(3, 7 / 30.0);
208             pv2.set(4, 1 / 15.0);
209             pv2.set(5, 0);
210
211             pv3.set(0, 1 / 15.0);
212             pv3.set(1, 1 / 15.0);
213             pv3.set(2, 7 / 30.0);
214             pv3.set(3, 2 / 5.0);
215             pv3.set(4, 1 / 15.0);
216             pv3.set(5, 1 / 6.0);
217
218             pv4.set(0, 1 / 5.0);
219             pv4.set(1, 1 / 5.0);
220             pv4.set(2, 1 / 5.0);
221             pv4.set(3, 1 / 5.0);
222             pv4.set(4, 1 / 5.0);
223             pv4.set(5, 0);
224
225             assertEquals(msv0, model2.getCoratingsVector(0, items));
226             assertEquals(msv1, model2.getCoratingsVector(1, items));
227             assertEquals(msv2, model2.getCoratingsVector(2, items));
228             assertEquals(msv3, model2.getCoratingsVector(3, items));
229             assertEquals(msv4, model2.getCoratingsVector(4, items));
230             assertEquals(msv5, model2.getCoratingsVector(5, items));
231             assertEquals(pv0, model2.getProximityVector(0, items));
232             assertEquals(pv1, model2.getProximityVector(1, items));
233             assertEquals(pv2, model2.getProximityVector(2, items));
234             assertEquals(pv3, model2.getProximityVector(3, items));
235             assertEquals(pv4, model2.getProximityVector(4, items));
236         }
237     }
```

HIRItemScorerTest.java
lenskit/lenskit-hir/src/test/java/org/lenskit/hir/HIRItemScorerTest.java

```
1  /**
2   * Tests HIR Item Scorer
3   */
4
5  public class HIRItemScorerTest {
6
7      @Rule
8      public TemporaryFolder folder = new TemporaryFolder();
9      MapItemGenreDAO gdao;
10     private static final double EPSILON = 1.0e-6;
11
12     @Before
13     public void createFile() throws IOException {
14         File f = folder.newFile("genres.csv");
15         PrintStream str = new PrintStream(f);
16         try {
17             str.println("0,\"Shawshank Redemption, The
                   (1994)\",0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
18             str.println("1,American History X
                   (1998),0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|0|0|0");
19             str.println("2,Z (1969),0|0|0|0|0|0|0|1|0|0|0|0|1|0|0|1|0|0|0");
20             str.println("3,\"Pan's Labyrinth (Laberinto del fauno, El)
                   (2006)\",0|0|0|0|0|0|0|1|1|0|0|0|0|0|1|0|0|0|0");
21             str.println("4,Seven Pounds
                   (2008),0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0");
22             str.println("5,Song of the Sea
                   (2014),0|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0");
23         } finally {
24
25             str.close();
26         }
27         gdao = MapItemGenreDAO.fromCSVFile(f);
28     }
29
30     @Test
31     public void testPredict1() throws RecommenderBuildException {
32
33         List<Rating> rs = new ArrayList<>();
34             rs.add(Rating.create(1, 0, 4));
35             rs.add(Rating.create(1, 4, 3));
36             rs.add(Rating.create(1, 5, 1));
37             rs.add(Rating.create(2, 0, 4));
38             rs.add(Rating.create(2, 4, 4));
39             rs.add(Rating.create(2, 5, 4));
40             rs.add(Rating.create(3, 0, 1));
41             rs.add(Rating.create(3, 4, 1));
42             rs.add(Rating.create(3, 5, 3));
43
44         Collection<Long> items = new HashSet<>();
45         items.add((long)0);
46         items.add((long)1);
47         items.add((long)2);
48         items.add((long)3);
49         items.add((long)4);
50         items.add((long)5);
51
52         ItemDAO idao = new ItemListItemDAO(LongUtils.packedSet(0, 1, 2, 3, 4, 5));
53         LenskitConfiguration config = new LenskitConfiguration();
54         config.bind(MapItemGenreDAO.class).to(gdao);
```

```
55          config.bind(EventDAO.class).to(EventCollectionDAO.create(rs));
56          config.bind(ItemDAO.class).to(idao);
57          config.bind(ItemScorer.class).to(HIRItemScorer.class);
58          config.bind(PreferenceDomain.class).to(new PreferenceDomainBuilder(0, 1)
59                                                      .setPrecision(1)
60                                                      .build());
61
62          ResultMap predictor1 = LenskitRecommenderEngine.build(config)
63                                                      .createRecommender(config)
64                                                      .getItemScorer()
65                                                      .scoreWithDetails(1,
                                                            items);
66          ResultMap predictor2 = LenskitRecommenderEngine.build(config)
67                                                      .createRecommender(config)
68                                                      .getItemScorer()
69                                                      .scoreWithDetails(2,
                                                            items);
70          ResultMap predictor3 = LenskitRecommenderEngine.build(config)
71                                                      .createRecommender(config)
72                                                      .getItemScorer()
73                                                      .scoreWithDetails(3,
                                                            items);
74
75          assert predictor1.size() == 3;
76          assert predictor2.size() == 3;
77          assert predictor3.size() == 3;
78
79
80  /*  Rank =
81           11/40       3/40       3/80      7/160      21/80     49/160
82          173/600     11/200      3/100      7/150     79/300      19/60
83          293/1000    33/1000     9/500      6/125     139/500     33/100
84  */
85
86          assertEquals(3.0 / 40.0, predictor1.getScore(1), EPSILON);
87          assertEquals(3.0 / 80.0, predictor1.getScore(2), EPSILON);
88          assertEquals(7.0 / 160.0, predictor1.getScore(3), EPSILON);
89          assertEquals(11.0 / 200.0, predictor2.getScore(1), EPSILON);
90          assertEquals(3.0 / 100.0, predictor2.getScore(2), EPSILON);
91          assertEquals(7.0 / 150.0, predictor2.getScore(3), EPSILON);
92          assertEquals(33.0 / 1000.0, predictor3.getScore(1), EPSILON);
93          assertEquals(9.0 / 500.0, predictor3.getScore(2), EPSILON);
94          assertEquals(6.0 / 125.0, predictor3.getScore(3), EPSILON);
95      }
96  }
```

build.gradle

lenskit/lenskit-hir/build.gradle

```
1   apply from: "$rootDir/gradle/maven.gradle"
2   apply from: "$rootDir/gradle/test-utils.gradle"
3
4   dependencies {
5       compile localGroovy()
6       compile project(':lenskit-core')
7       compile project(':lenskit-knn')
8   }
9
10  meta {
11      name 'LensKit HIR'
12      description 'HIR recommender for LensKit'
13  }
```

retrieveGenres.py

```
import csv
new_rows = []
changes = {
        'Action' : 524288,
        'Adventure' : 262144,
        'Animation' : 131072,
        'Children' : 65536,
        'Comedy'   : 32768,
        'Crime' : 16384,
        'Documentary' : 8192,
        'Drama' : 4096,
        'Fantasy' : 2048,
        'Film-Noir' : 1024,
        'Horror' : 512,
        'Musical' : 256,
        'Mystery' : 128,
        'Romance' : 64,
        'Sci-Fi' : 32,
        'Thriller' : 16,
        'War' : 8,
        'Western' : 4,
        'IMAX' : 2,
        '(no genres listed)' : 1
        }

with open('genres.csv', 'r') as f:
        spamreader = csv.reader(f, delimiter=',', quotechar='"')
        for row in spamreader:
                deck =  [changes[x] for x in row[-1].split("|")]
                s = sum(deck)
                s = '{0:20b}'.format(s)
                row[-1] = "|".join(s)
                row[-1] = row[-1].replace(' ','0')
                new_rows.append(row) # add the modified rows

with open('genres.csv', 'w') as f:
        # Overwrite the old file with the modified rows
        writer = csv.writer(f)
        writer.writerows(new_rows)
```

range.py

```
import csv
new_rows = []

with open('genres.csv', 'r') as f:
        spamreader = csv.reader(f, delimiter=',', quotechar='"')
        i = 0
        for row in spamreader:
                row[0] = i
                new_rows.append(row) # add the modified rows
                i = i + 1

with open('genres.csv', 'w') as f:
        # Overwrite the old file with the modified rows
        writer = csv.writer(f)
        writer.writerows(new_rows)


newRows = []
with open('movies.csv', 'r') as f:
```

```
        spamreader = csv.reader(f, delimiter=',', quotechar='"')
        i = 0
        for row in spamreader:
                row[0] = i
                newRows.append(row) # add the modified rows
                i = i + 1

with open('movies.csv', 'w') as f:
        # Overwrite the old file with the modified rows
        writer = csv.writer(f)
        writer.writerows(newRows)
```

ratings.py

```
import csv
new_rows = []

d = {}

with open('forids.csv', 'r') as f:
        spamreader = csv.reader(f, delimiter=',', quotechar='"')
        i = 0
        for row in spamreader:
                d[row[0]] = i
                i = i + 1

with open('ratings.csv', 'r') as f:
        spamreader = csv.reader(f, delimiter=',', quotechar='"')
        for row in spamreader:
                row[1] = d[row[1]]
                new_rows.append(row) # add the modified rows

with open('ratings.csv', 'w') as f:
        # Overwrite the old file with the modified rows
        writer = csv.writer(f)
        writer.writerows(new_rows)
```

testFun.m

```
A = [ 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 ; 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
     0 0 0 ; 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 ; 0 0 0 0 0 0 0 1 1 0 0 0 0
     0 0 1 0 0 0 0 ; 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 ; 0 0 1 1 0 0 0 0 1
     0 0 0 0 0 0 0 0 0 0 0 0 ];
numRowsA = size(A, 1);
S = sum(A, 2)

for i = 1:numRowsA

        R(i, :) = A(i, :) / S(i);

end

Y = transpose(A)


numRowsY = size(Y, 1)

SY = sum(Y, 2)

for i = 1:numRowsY

        if SY(i) != 0

                T(i, :) = Y(i, :) / SY(i);
```

```
23
24          else T(i, :) = Y(i, :);
25
26          end
27
28  end
29
30  P = R * T;
31
32  PF = [4 0 0 0 3 1 ;
33        4 0 0 0 4 4 ;
34        1 0 0 0 1 3]
35
36  numRowsPFV = size(PF, 1)
37
38  SPFV = sum(PF, 2)
39
40  for i = 1:numRowsPFV
41
42          if PF(i) != 0
43
44                  PF(i, :) = PF(i, :) / SPFV(i);
45
46          else PF(i, :) = PF(i, :);
47
48          end
49
50  end
51
52  C = [0 0 0 0 0.5 0.5 ;
53       0 0 0 0  0   0  ;
54       0 0 0 0  0   0  ;
55       0 0 0 0  0   0  ;
56     0.5 0 0 0  0  0.5 ;
57     0.5 0 0 0 0.5  0 ];
58
59
60
61  a = 0.1
62
63  CR = 0.6*C
64
65  P = 0.3*P
66
67  CRP = CR + P
68
69  Rank = a*PF + PF*CRP
```

Κεφάλαιο 7

Ορολογία

Ξενόγλωσσος Όρος	Ελληνικός Όρος
APIs	Διεπαφές Προγραμματισμού Εφαρμογών
Baseline Scorer	Βαθμολογητής Βάσης
batch	μαζικός
Batch Like	Μαζική Προτίμηση
builder	κατασκευαστής
Builder and Facade	Κατασκευή και Εκπροσώπηση
cold start	κρύα εκκίνηση
Collaborative Filtering	Συνεργατική Διήθηση
collection	συλλογή
collection API	Διεπαφή συλλογών
Comma-Separated Values	Τιμές Διαχωρισμένες με Κόμμα
Command Line interface	Διεπαφή Γραμμής Εντολών
community-based	βάσει κοινότητας
component	τμήμα
configuration files	αρχεία διαμόρφωσης
Content Analyzer	Αναλυτής Περιεχομένου
content-based	βάσει περιεχομένου
core	πυρήνας
cross-validation	διαχωρισμού-επαλήθευσης
damping parameter	παράμετρος απόσβεσης
DAO	Αντικείμενο για την πρόσβαση σε δεδομένα
data container	τμήμα που περιέχει δεδομένα
Data Structures	Δομές Δεδομενων
Degree of Agreement	Βαθμός Συμφωνίας

Συνέχεια στην επόμενη σελίδα

Συνέχεια από τη προηγούμενη σελίδα

Ξενόγλωσσος Όρος	**Ελληνικός Όρος**
delimited files	οριοθετημένα αρχεία
demographic	δημογραφικός
Dependency injection	Εισαγωγή εξαρτήσεων
dependency injectors	εισαγωγείς εξαρτήσεων
Entropy	Εντροπία
Evaluator	Αξιολογητής
Event	Γεγονός
fast iteration	γρήγορη επανάληψη
Filtering Component	Τμήμα Διήθησης
framework	πλαίσιο εργασίας
Gaussian Probabilistic Latent Semantic Analysis	Γκαουσιάνη Πιθανοτική Λανθάνουσα Σημασιολογική Ανάλυση
hash map	πίνακας κατακερματισμού
history summarizer	παραγωγός περιλήψης ιστορικού
HIR	Ιεραρχική Κατάταξη βάσει του χώρου των ειδών
hybrid	υβριδικός
ID	αναγνωριστικό
immutable object	αμετάβλητο αντικείμενο
Immutable Sparse Vectors	Αμετάβλητα Αραία Διανύσματα
Inversion of Control	Αντιστροφή Ελέγχου
item	είδος
item-based	βασιζόμενο στα είδη
Item Recommender	Παραγωγός Συστάσεων Ειδών
Item Scorer	Βαθμολογητής Ειδών
key domain	χώρος κλειδιών
key set	σύνολο κλειδιών
knowledge-based	βάσει γνώσης
latent characteristic	λανθάνον χαρακτηριστικό
latent factor	λανθάνων παράγοντας
length	μήκος
Like	Προτίμηση
map	αντιστοίχιση
matrix factorizaion	παραγοντοποίηση μητρώου
Mean absolute error	Μέσο απόλυτο σφάλμα
model based	βασιζόμενη στο μοντέλο
module	ενότητα
MOOC	Μαζικό ανοιχτό διαδικτυακό μάθημα

Συνέχεια στην επόμενη σελίδα

Συνέχεια από τη προηγούμενη σελίδα

Ξενόγλωσσος Όρος	**Ελληνικός Όρος**
Mutable Sparse Vectors	Μεταβλητά Αραία Διανύσματα
neighborhood-based	βάσει γειτονιάς
New Community Problem	Πρόβλημα Νέας Κοινότητας
New Items Problem	Πρόβλημα Νέων Ειδών
New Users Problem	Πρόβλημα Νέων Χρηστών
Normalized Distance-based Performance Measure	Μετρική Κανονικοποιημένης Απόδοσης βάσει Απόστασης
normalizer	κανονικοποιητής
numeric identifiers	αριθμητικά αναγνωριστικά
offline	εκτός σύνδεσης
pack	δέσμη
path	μονοπάτι
popularity	δημοφιλία
pre-built	προκατασκευασμένα
precision	ακρίβεια
predictor	παραγωγός προβλέψεων
Profile Learner	Κατασκευαστής Προφίλ
Ranking	Κατάταξη
Rating	Βαθμολογία χρήστη
Rating Predictor	Παραγωγός Πρόβλεψης Βαθμολογίας
recall	ανάκληση
Recommender	Παραγωγός Συστάσεων
Root mean squared error	Μέσο τετραγωγικό σφάλμα
subsample	υπο-δείγμα
score	βαθμολογία είδους
script	σενάριο
Sparse Vectors	Αραιά Διανύσματα
sparsity	αραιότητα
Strategy Pattern	Μοτίβο στρατηγικής
streaming	συνεχούς ροής
SVD	Παραγοντοποίηση Ιδιαζουσών Τιμών
tag	ετικέτα
train/test	εκπαίδευσης και ελέγχου
toolkit	εργαλειοθήκη
user	χρήστης
user-based	βασιζόμενο στους χρήστες
vector	διάνυσμα

Κεφάλαιο 8

Συντμήσεις-Αρκτικόλεξα

Σύντμηση	Πλήρης Ανάπτυξη
API	Application programming interface
CLI	Command Line interface
CSV	Comma-Separated Values
DAO	Data Access Object
DOA	Degree of Agreement
HIR	Hierarchical Itemspace Rank
HPPC	High Performance Primitive Collections for Java
MAE	Mean Absolute Error
MAP	mean average precision
MRR	Mean reciprocal rank
MOOC	Massive Open Online Course
nDCG	Normalized discounted cumulative gain
NDPM	Normalized Distance-based Performance Measure
RMSE	Root mean squared error
SVD	Singular Value Decomposition

Βιβλιογραφία

[1] Bloch, Joshua. *Effective Java (2Nd Edition) (The Java Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2008, ISBN 0321356683, 9780321356680.

[2] Burke, Robin. *The adaptive web*. chapter Hybrid Web Recommender Systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg, 2007, ISBN 978-3-540-72078-2. http://dl.acm.org/citation.cfm?id=1768197.1768211.

[3] Chebotarev, P and E Shamis. *The Matrix-Forest Theorem and Measuring Relations in Small Social Groups*. Autom. Remote Control., 58(math.CO/0602070. 9):1505–1514. 10 p, Feb 2006. http://cds.cern.ch/record/927673.

[4] Cremonesi, Paolo, Yehuda Koren, and Roberto Turrin. *Performance of recommender algorithms on top-n recommendation tasks*. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM, ISBN 978-1-60558-906-0. http://doi.acm.org/10.1145/1864708.1864721.

[5] Ekstrand, Michael D.. *Towards Recommender Engineering: Tools and Experiments in Recommender Differences*. Ph.d thesis, University of Minnesota, Minneapolis, MN, July 2014. http://md.ekstrandom.net/research/thesis/.

[6] Ekstrand, Michael D., Michael Ludwig, Joseph A. Konstan, and John T. Riedl. *Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit*. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0683-6. http://doi.acm.org/10.1145/2043932.2043958.

[7] Fouss, Francois, Alain Pirotte, Jean Michel Renders, and Marco Saerens. *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation*. IEEE Trans. on Knowl. and

Data Eng., 19(3):355–369, March 2007, ISSN 1041-4347. http://dx.doi.org/10.1109/TKDE.2007.46.

[8] Fouss, François, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. *An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification*. Neural Networks, 31:53 – 72, 2012, ISSN 0893-6080. http://www.sciencedirect.com/science/article/pii/S0893608012000822.

[9] Furnell, S.. *Proceedings of the Second International Network Conference (INC2000)*. University of Plymouth, 2000, ISBN 9781841020662. https://books.google.gr/books?id=OLrZAwAAQBAJ.

[10] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995, ISBN 0-201-63361-2.

[11] Good, Nathaniel, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. *Combining collaborative filtering with personal agents for better recommendations*. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence, ISBN 0-262-51106-1. http://dl.acm.org/citation.cfm?id=315149.315352.

[12] Gori, Marco and Augusto Pucci. *Itemrank: A random-walk based scoring algorithm for recommender engines*. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2766–2771, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=1625275.1625720.

[13] Hofmann, Thomas. *Collaborative filtering via gaussian probabilistic latent semantic analysis*. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM, 2003.

[14] Katz, Leo. *A new status index derived from sociometric analysis*. Psychometrika, 18(1):39–43, March 1953. http://ideas.repec.org/a/spr/psycho/v18y1953i1p39-43.html.

[15] Kendall, M.. *Rank Correlation Methods*. Charles Griffin & Company Limited, 1948.

[16] Konstan, Joseph A., J. D. Walker, D. Christopher Brooks, Keith Brown, and Michael D. Ekstrand. *Teaching recommender systems at large scale: Evaluation and lessons learned from a hybrid mooc*. ACM Trans. Comput.-Hum. Interact., 22(2):10:1–10:23, April 2015, ISSN 1073-0516. http://doi.acm.org/10.1145/2728171.

[17] Koren, Yehuda. *Factorization meets the neighborhood: A multifaceted collaborative filtering model*. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-193-4. http://doi.acm.org/10.1145/1401890.1401944.

[18] Martin, Robert C.. *The dependency inversion principle*. C++ Report, 8, May 1996.

[19] Masthoff, Judith. *Modeling the multiple people that are me*. In *User Modeling 2003, 9th International Conference, UM 2003, Johnstown, PA, USA, June 22-26, 2003, Proceedings*, pages 258–262, 2003. http://dx.doi.org/10.1007/3-540-44963-9_34.

[20] McDonald, John H.. *Handbook of Biological Statistics*. Sparky House Publishing, Baltimore, Maryland, USA, third edition, 2014. http://udel.edu/~mcdonald/statintro.html.

[21] Mitchell, Thomas M.. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1st edition, 1997, ISBN 0070428077, 9780070428072.

[22] Mladenic, Dunja. *Text-learning and related intelligent agents: A survey*. IEEE Intelligent Systems, 14(4):44–54, July 1999, ISSN 1541-1672. http://dx.doi.org/10.1109/5254.784084.

[23] Nikolakopoulos, Athanasios N. and John D. Garofalakis. *NCDawareRank: a novel ranking method that exploits the decomposable structure of the web*. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 143–152, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-1869-3. http://doi.acm.org/10.1145/2433396.2433415.

[24] Nikolakopoulos, Athanasios N., Marianna A. Kouneli, and John D. Garofalakis. *Hierarchical itemspace rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation*. Neurocomputing, 163(1):126 – 136, 2015, ISSN 0925-2312. http://www.sciencedirect.com/science/article/pii/S0925231215002180, Recent Advancements in Hybrid Artificial Intelligence Systems and its Application to Real-World ProblemsProgress in Intelligent SystemsMining Humanistic DataSelected papers from the 7th International Conference on Hybrid Artificial Intelligence Systems (HAIS

2012)Selected papers from the 2nd Brazilian Conference on Intelligent Systems (BRACIS 2013)Selected papers from the 2nd Mining Humanistic Data Workshop at {EANN} 2013.

[25] Ricci, Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender systems handbook*. Springer, 2011.

[26] Siegel, S. and N. J. Castellan. *Non Parametric Statistics for the Behavioral Sciences*. McGraw-Hill International, 2nd edition, 1988.

[27] Sinha, Rashmi R. and Kirsten Swearingen. *Comparing recommendations made by online systems and friends*. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001. http://dblp.uni-trier.de/db/conf/delos/delos2001.html#SinhaS01.

[28] Takács, Gábor, István Pilászy, Bottyán Németh, and Domonkos Tikk. *Major components of the gravity recommendation system*. SIGKDD Explor. Newsl., 9(2):80–83, December 2007, ISSN 1931-0145. http://doi.acm.org/10.1145/1345448.1345466.

[29] Yao, Y. Y.. *Measuring retrieval effectiveness based on user preference of documents*. J. Am. Soc. Inf. Sci, 1995.