POLITECNICO DI MILANO
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Analisi e correzione dell'eccessiva dinamicità delle liste di raccomandazione

Laboratorio di Valutazione delle Prestazioni

Relatore: Ing. Roberto Turrin
Correlatore: Prof. Paolo Cremonesi

Tesi di Laurea di:
Matteo Restelli Matr. n. 724712
Martino Zocca Matr. n. 724755

Anno Accademico 2009-2010

A chi ci è stato vicino...

Sommario

I provider televisivi che offrono servizi IPTV e Video on Demand presentano ormai cataloghi con migliaia di film e programmi televisivi. Di conseguenza molto contenuti sono poco visibili e la ricerca da parte dell'utente di materiale interessante è difficoltosa. In aggiunta, il tipo di interazione dell'utente con il sistema IPTV è tipicamente effettuata tramite telecomando e presenta evidenti limitazioni.

Per questo motivo diversi provider stanno integrando, nella piattaforma IPTV, un sistema di raccomandazione in grado di fornire agli utenti delle liste di contenuti personalizzati in base alle loro preferenze. Tali liste di raccomandazione possono variare, anche notevolmente, tra le varie sessioni di interazione dell'utente con il sistema. Questa dinamicità, se eccessiva, porta l'utente a perdere fiducia verso il sistema, pensando addirittura che gli vengano consigliati dei film casualmente.

In questa tesi viene (i) analizzata la dinamicità delle liste di raccomandazione basandosi su due dataset reali e (ii) viene presentato un possibile metodo di correzione della dinamicità per controllare variazioni.

Ringraziamenti

Ringraziamo Roberto e il professor Cremonesi che ci hanno permesso di lavorare in un campo di ultima generazione. In particolare Roberto che ci aiutato e sopportato in questi ultimi due anni.

Matteo: vorrei ringraziare tutte le persone che mi sono state vicine in questo periodo universitario. La mia famiglia, i miei compagni universitari (Alessandro, Alex,Daniele e Fernanda, Mango, Martino e Cristina, Marco, Mattia, Ricky Wezza, Ricky biondo, Rocco, Ivan), i ragazzi del VPLAB (Boe, Fabio, Luca, Pietro), i miei amici (Anna, Bosh, Braizio, Chiara, Faro, Flisi, Lalla, Limo, Master, Milo, Moore, Teo Picchi, Ste), tutta Galgiana (Cesare compreso), lo staff della Latteria (Marco, Mirella, Martina) ed eventualmente tutti quelli che mi son dimenticato di inserire qui. Grazie di cuore.

Martino: Ringrazio il mio compagno di tesi Matteo perchè ce la siamo spassata per bene in tutti questi anni, e adesso è anche comprensibile che non mi voglia più vedere per un bel po'. Ringrazio Marco, Ricky e Tia che mi son sempre stati vicino. Ringrazio il VPLAB che mi ha ospitato a 20°C per quest'ultimo anno e in particolare Pietro, Luca e Antonella. Ringrazio chi ce l'ha fatta prima di me (Lenny,Blondie,Ale,Ivan,Alex), ringrazio Bocco e Vermezzo. Ringrazio la famiglia Albertini che mi ha sfamato in tutti questi anni. Ringrazio la mia famiglia che mi ha permesso tutto questo, supportandomi, e ringrazio Cristina perchè alla fine tutto questo, spero, sarà per noi.

Indice

Sommario **I**

Ringraziamenti **III**

Elenco delle figure **1**

Elenco delle tabelle **4**

1 Introduzione **7**
1.1 Contributi e risultati principali 8
1.2 Struttura della tesi 9

2 Stato dell'arte **11**
2.1 Introduzione 11
2.2 Sistemi di Raccomandazione 13
2.3 Dataset 14
2.3.1 Rating Espliciti 16
2.3.2 Rating Impliciti 16
2.3.3 Evoluzione temporale di un dataset 17
2.4 Architettura di un sistema di raccomandazione 18
2.5 Algoritmi di raccomandazione 19
2.5.1 Algoritmi Content-Based 20
2.5.2 Algoritmi Collaborativi 24
2.6 Lavori Correlati 30

3 Algoritmi di raccomandazione utilizzati **33**
3.1 Introduzione 33
3.2 Algoritmi con modello nello spazio latente 33
3.3 Algoritmi Item-Item Cosine KNN 34
3.4 Algoritmo Item-Item DR 36
3.5 Algoritmo Sarwar 36

3.6 Algoritmo Asymmetric SVD 38
3.6.1 Modifiche apportate all'algoritmo Asymmetric SVD . 39
3.7 Algoritmo Content Based: LSA Cosine 39
3.7.1 Fase Batch: generazione del modello 39
3.7.2 Fase Real-Time: generazione della raccomandazione . 40

4 Descrizione del problema e verifica esistenza 43
4.1 Descrizione del problema 43
4.2 Dataset 46
4.2.1 Dataset statico MovieRec 46
4.2.2 Dataset statico Netflix 48
4.2.3 Dataset temporale MovieRec 50
4.3 Verifica dell'esistenza del problema 53
4.3.1 Prima metodologia di verifica 53
4.3.2 Risultati prima metodologia di verifica 56
4.3.3 Seconda metodologia di verifica: profili ad hoc 61
4.3.4 Risultati seconda metodologia di verifica: profili ad hoc 61

5 Framework Antireshuffling 69
5.1 Metodologia di risoluzione 69
5.2 Metriche di valutazione 72
5.2.1 Gap Posizione Assoluta 72
5.2.2 Gap Posizione Percentuale 73
5.3 Esempi di funzionamento 74
5.3.1 Gap Posizione Assoluta 74
5.3.2 Gap Posizione Percentuale 77

6 Metodologia di test e risultati 81
6.1 K-Fold Cross Validation 81
6.2 Bootstrap 82
6.3 Metriche di qualità 82
6.3.1 Recall 83
6.3.2 Precision 84
6.3.3 EPR 85
6.3.4 ARHR 85
6.4 Prima metodologia: test statistici 86
6.5 Risultati test statistici 89
6.5.1 Risultati riassuntivi dei test statistici 97
6.6 Risultati test sui profili ad hoc 101

7 Integrazione in un sistema reale **113**
7.1 L'evoluzione temporale 113
7.2 Evoluzione temporale del modello 116
7.2.1 Risultati evoluzione temporale del modello 117
7.3 Evoluzione temporale dei profili utente 120
7.3.1 Risultati evoluzione temporale dei profili utente 121
7.4 Evoluzione temporale del sistema di raccomandazione 123
7.4.1 Risultati evoluzione temporale del sistema di raccomandazione 124

8 Conclusioni e sviluppi futuri **129**

Bibliografia **131**

Elenco delle figure

2.1 Matrice ICM 15
2.2 Matrice URM 15
2.3 Fasi di batch e real-time 19
2.4 Classificazione algoritmi di raccomandazione 20
2.5 Procedimento algoritmo Content-Based diretto 22
2.6 Vettori Utente e Item 23
2.7 Calcolo della similarità tra item 27
2.8 Esempio vettori utente 28
2.9 Matrice ricavata dai vettori utente 28
2.10 Esempio vettori oggetti 28
2.11 Matrice ricavata dai vettori oggetti 28

3.1 Matrice di similarità dell'Item-Item Cosine 36
3.2 Costruzione del modello per l'algoritmo Sarwar e matrice di similarità tra gli item 37
3.3 Matrice di similarità dell'algoritmo Asymmetric SVD 38
3.4 Costruzione del modello per l'algoritmo LSA e matrice di similarità tra gli item 40

4.1 Caratteristiche dataset utilizzati 46
4.2 Dataset MovieRec: distribuzione utenti per lunghezza del profilo 47
4.3 Dataset MovieRec: percentuale di voti in base alla percentuale di film 48
4.4 Dataset Netflix: distribuzione utenti per lunghezza del profilo 48
4.5 Dataset Netflix: distribuzione utenti per lunghezza del profilo al massimo di 20 rating 49
4.6 Dataset Netflix: percentuale di voti in base alla percentuale di film 49
4.7 Dataset MovieRec temporale: distribuzione utenti per lunghezza del profilo 50

4.8 Dataset MovieRec temporale: percentuale di voti in base alla percentuale di film 51
4.9 Frequeza settimanale di aggiunta dei rating nel dataset 52
4.10 Frequeza mensile di aggiunta dei rating nel dataset 52
4.11 Dataset MovieRec: percentuali utenti per categoria 54
4.12 Dataset Netflix: percentuali utenti per categoria 54
4.13 Dataset MovieRec - Algoritmo Sarwar: Valore di Reshuffling per profilo di partenza popolare (a) e non popolare (b) 57
4.14 Dataset Netflix - Algoritmo Asymmetric SVD: Valore di Reshuffling per profilo di partenza popolare (a) e non popolare (b) . 57
4.15 Dataset MovieRec: Valore di Reshuffling per la prima categoria per profilo di partenza popolare (a) e non popolare (b) 58
4.16 Dataset MovieRec: Valore di Reshuffling per la seconda categoria per profilo di partenza popolare (a) e non popolare (b) 58
4.17 Dataset Netflix: Valore di Reshuffling per la prima categoria per profilo di partenza popolare (a) e non popolare (b) 59
4.18 Dataset Netflix: Valore di Reshuffling per la seconda categoria per profilo di partenza popolare (a) e non popolare (b) 59

5.1 Workflow di funzionamento dell'algoritmo correttivo 71
5.2 Gap Posizione Assoluta - Passo 1: Lista al tempo $t1$ (sinistra) e lista al tempo $t2$(destra) 74
5.3 Gap Posizione Assoluta - Passo 2: Lista entranti (sinistra) e lista uscenti $t2$(destra) 75
5.4 Gap Posizione Assoluta - Passo 3: Lista entranti (sinistra) e lista uscenti $t2$(destra) con relativo gap 75
5.5 Gap Posizione Assoluta - Passo 4: Lista entranti (sinistra) e lista uscenti $t2$(destra) dopo il primo filtraggio 76
5.6 Gap Posizione Assoluta - Passo 5: Lista entranti (sinistra) e lista uscenti $t2$(destra) dopo il secondo filtraggio 76
5.7 Gap Posizione Assoluta - Passo 6: Lista di raccomandazione restituita dall'algoritmo correttivo 77
5.8 Gap Posizione Percentuale - Passo 1: Lista al tempo $t1$ (sinistra) e lista al tempo $t2$(destra) 77
5.9 Gap Posizione Percentuale - Passo 2: Lista entranti (sinistra) e lista uscenti $t2$(destra) 78
5.10 Gap Posizione Percentuale - Passo 3: Lista entranti (sinistra) e lista uscenti $t2$(destra) con relativo gap 78

5.11 Gap Posizione Percentuale - Passo 4: Lista entranti (sinistra) e lista uscenti $t2$(destra) dopo il primo filtraggio 78
5.12 Gap Posizione Percentuale - Passo 5: Lista entranti (sinistra) e lista uscenti $t2$(destra) dopo il secondo filtraggio 79
5.13 Gap Posizione Percentuale - Passo 6: Lista di raccomandazione restituita dall'algoritmo correttivo 79

6.1 Esempio metodo K-Fold Cross Validation 82
6.2 Esempio metodo Bootstrap 83
6.3 Modifiche effettuate al profilo durante il test 87
6.4 Worflow Test Statistico: Ten Fold Cross Validation 88
6.5 Caratteristiche Boxplot 89
6.6 Dataset MovieRec - Algoritmo Sarwar - Metrica Gap Posizione Assoluta: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b) 90
6.7 Dataset MovieRec - Algoritmo Sarwar - Metrica Gap Posizione Percentuale: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b) 90
6.8 Dataset Netflix - Algoritmo AsymmetricSVD - Metrica Gap Posizione Assoluta: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b) 91
6.9 Dataset Netflix - Algoritmo AsymmetricSVD - Metrica Gap Posizione Percentuale: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b 91
6.10 Grafici Riassuntivi con Confronto Metriche per il dataset MovieRec (a) e Netflix (b) 92
6.11 Legenda 92
6.12 Dataset MovieRec - Recall / Precision per la metrica Gap Posizione Assoluta (a) e Percentuale (b) (L'indicatore è posto a N=10 95
6.13 Dataset Netflix - Recall / Precision per la metrica Gap Posizione Assoluta (a) e Percentuale (b) (L'indicatore è posto a N=10) 95

7.1 Dataset MovieRec Temporale: Evoluzione dei film e dei ratings 116
7.2 Evoluzione temporale del modello con profili fissi 117
7.3 Evoluzione temporale del modello, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 118
7.4 Evoluzione temporale dei profili con modello fisso 120

7.5 Evoluzione temporale dei profili utente, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 122
7.6 Evoluzione temporale del sistema di raccomandazione 123
7.7 Evoluzione temporale del sistema di raccomandazione, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 125

Elenco delle tabelle

4.1 Dataset MovieRec: risultati profili ad hoc (in grigio le aspettative non verificate) 63
4.2 Dataset MovieRec: risultati profili ad hoc (in grigio le aspettative non verificate) 64
4.3 Dataset Netflix: risultati profili ad hoc (in grigio le aspettative non verificate) 66
4.4 Dataset Netflix: risultati profili ad hoc (in grigio le aspettative non verificate) 67

6.1 Tabelle Riassuntive Metriche di Qualità per i dataset MovieRec(a) e Netflix(b) 97
6.2 Dataset MovieRec: Tabelle Metriche di Qualità al variare della soglia per la metrica Gap Posizione Assoluta (a) e Percentuale (b) con l'algoritmo Sarwar 98
6.3 Dataset Netflix: Tabelle Metriche di Qualità al variare della soglia per la metrica Gap Posizione Assoluta (a) e Percentuale (b) con l'algoritmo Asymmetric SVD 98
6.4 Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 103
6.5 Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 104
6.6 Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 105
6.7 Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 107
6.8 Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 108
6.9 Dataset Netflix : risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 109
6.10 Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate) 110

7.1 Evoluzione temporale del modello, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 117
7.2 Evoluzione temporale dei profili utente, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 121
7.3 Evoluzione temporale dei sistema di raccomandazione, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b) 124

Capitolo 1

Introduzione

La diffusione di tecnologie quali IPTV (IP Television) e VoD (Video on Demand) ha modificato il modo di concepire la televisione, che è sempre più uno strumento *interattivo*. Se da un lato l'utente dispone di un numero sempre crescente di contenuti disponibili, la ricerca e la selezione di tali contenuti è notoriamente scomoda. Cataloghi di grandi dimensioni devono infatti essere sfogliati attraverso l'utilizzo di un classico telecomando. Le conseguenze sono sia una scarsa visibilità dei contenuti meno popolari sia l'insoddisfazione degli utenti che non riescono a trovare materiale interessante.

Diversi IPTV/VoD provider, consapevoli del problema, stanno sperimentando l'integrazione di **sistemi di raccomandazione**. Tali sistemi sono in grado, a real-time, di suggerire un numero limitato ma molto specifico di contenuti altamente *personalizzati* per ciascun utente, così da soddisfare le preferenze degli utenti e incrementare, di conseguenza, le vendite.

Un sistema di raccomandazione è in grado di personalizzare i contenuti proposti sulla base (i) delle preferenze degli utenti - rappresentate dal cosiddetto *profilo utente* - e (ii) delle caratteristiche dei contenuti da proporre (modello degli item).

Il profilo utente consiste in una serie di informazioni sull'interazione tra l'utente stesso e film o programmi presenti a catalogo. In sintesi, rappresenta una sorta di storico di quali contenuti ha visionato l'utente all'interno del sistema e di quanto questi contenuti sono stati di suo gradimento.

Il modello degli item rappresenta le caratteristiche principali dei film o programmi televisivi presenti a catalogo. A seconda dell'algoritmo di raccomandazione utilizzato, tali caratteristiche possono riguardare il contenuto (es., attori, regista,...) - algoritmi Content-Based - oppure le preferenze della comunità di utenti - algoritmi Collaborativi.

Il sistema di raccomandazione in base a queste informazioni - profilo utente e modello degli item - decide quali film raccomandare all'utente stesso, proponendo una lista di N film (**lista di raccomandazione Top-N**), dove N è tipicamente tra 5 e 10. L'obiettivo del sistema di raccomandazione è ovviamente quello proporre in questa lista contenuti di alto interesse per l'utente.

Per un dato utente, la lista di raccomandazione Top-N può cambiare o per una modifica al suo profilo utente o per una variazione del modello degli item. Se da un lato proporre continuamente nuovi item potrebbe rappresentare un fattore positivo, indicando che il sistema agisce effettivamente a real-time consigliando contenuti specifici e aggiornati con l'attuale profilo dell'utente, dall'altro una troppa **dinamicità della lista di raccomandazione** può sfiduciare l'utente nei confronti del sistema (all'estremo, l'utente potrebbe pensare che esso consigli contenuti casualmente).

Un cambiamento dovrebbe essere completamente giustificato da nuove preferenze dell'utente, in modo che quest'ultimo possa capire perché tutto o parte di quello che gli veniva raccomandato precedentemente non sia più presente. Se l'utente invece, rimane sorpreso da tutta questa diversità tra la lista di raccomandazione attuale e quella a lui precedentemente presentata, il cambiamento può essere conseguenza di una scarsa robustezza della raccomandazione e quindi divenire un problema.

L'obiettivo di questo lavoro di tesi è quello di (i) analizzare l'eccessiva dinamicità delle liste di raccomandazione presente in dataset reali e (ii) proporre una metodologia correttiva per alleviare il problema. Tale soluzione deve eventualmente ridurre la dinamicità, mantenendo comunque una raccomandazione adeguata in termini di qualità.

1.1 Contributi e risultati principali

Il lavoro si è basato su due dataset reali e molteplici algoritmi di raccomandazione noti in letteratura.

Inizialmente è stata effettuata una valutazione sull'effettiva presenza del problema, attraverso l'uso di test statistici off-line, in cui vengono analizzati con meccanismi automatici le liste di raccomandazione. In aggiunta a questi test sono stati creati manualmente una serie di profili con diverse caratteristiche in modo da testare il problema in maniera soggettiva.

I test statistici hanno mostrato che la maggior parte degli algoritmi, su tutti i dataset utilizzati, era affetta da questo problema mediamente in percentuali dal 20% al 60%. I test personalizzati hanno mostrato che le aspet-

tative soggettive (cioè il valore di dinamicità atteso dall'utente), soprattutto in uno dei dataset, non erano soddisfatte assolutamente.

Una volta verificata l'eccessiva presenza del problema è stato necessario analizzare i meccanismi con cui limitare la dinamicità delle liste. Anziché intervenire direttamente all'interno degli algoritmi di raccomandazione, cosa che sarebbe stata praticamente impossibile dato l'alto numero di algoritmi in letteratura ed al continuo proliferarsi di nuovi e più accurati algoritmi, si è preferito progettare un metodo che potesse valutare ogni singola modifica alla lista di raccomandazione (in pratica nuovi item che vengono consigliati) e in qualche modo giustificarla, eliminando dalla lista gli item consigliati ingiustificati a favore di quelli presenti nella lista precedente, in modo da diminuire la diversità tra una lista e la sua antecedente.

Per fare questo è stato sviluppato un algoritmo che agisce a valle della raccomandazione, il quale ricevendo come input la lista attuale e quella precedente, valuta la diversità tra di esse e per ogni singolo item cambiato, in base ad alcune metriche, decide se questo possa rimanere nella nuova lista da presentare all'utente.

Successivamente si è proceduto alla valutazione dell'algoritmo di correzione sviluppato, sulla base dei test preliminari effettuati, controllando tramite delle metriche per la valutazione della raccomandazione che l'algoritmo correttivo, in coda a quello di raccomandazione non ne vada a peggiorare le qualità correggendo le liste di raccomandazione prodotte. In questa fase di test si è mostrato che l'algoritmo correttivo proposto consente una diminuzione della dinamicità media delle liste di raccomandazione, senza eccessiva perdita di qualità - misurata tramite tipiche metriche di classificazione quali la Recall. Nei test statistici condotti la dinamicità è passata dal $20\% \div 60\%$ originale al $10\% \div 30\%$ con l'applicazione del metodo correttivo. I test personalizzati, hanno mostrato a loro volta l'effettivo funzionamento, dimostrato dall'incremento nel numero di aspettative soggettive verificate.

In aggiunta a questi test è stata effettuata una sorta di simulazione di integrazione dell'algoritmo correttivo attraverso l'uso di un dataset con informazioni temporale, con cui è stato possibile ricostruire la reale evoluzione del sistema in diversi istanti di tempo.

1.2 Struttura della tesi

La tesi è strutturata nel seguente modo.

Nel Capitolo 2 viene presentato il dominio dei sistemi di raccomandazione, gli algoritmi utilizzati da questi ultimi e i principali tipi di dati che vengono gestiti.

Nel Capitolo 3 vengono descritti in dettaglio gli algoritmi di raccomandazione presenti in letteratura che sono stati utilizzati nel corso di questo lavoro.

Nel Capitolo 4 si introduce il problema, si presentano i dataset utilizzati in fase di test, vengono descritte le metodologie di test mostrando i risultati ottenuti.

Nel Capitolo 5 viene descritta la metodologia sviluppata per la correzione del problema con particolare attenzione alle due metriche utilizzate nella fase di test.

Nel Capitolo 6 vengono descritte le metodologie di test e i risultati ottenuti dall'algoritmo correttivo, in riferimento a ciò che si è ottenuto nel Capitolo 4.

Nel Capitolo 7 viene mostrata una simulazione di applicazione dell'algoritmo correttivo in un contesto reale.

Nel Capitolo 8 vengono riportate le considerazioni finale sul lavoro effettuato e i possibili sviluppi futuri.

Capitolo 2

Stato dell'arte

In questo capitolo vengono presentati i sistemi di raccomandazione, la loro architettura, i dati che essi utilizzano e gli algoritmi che stanno alla base del loro funzionamento.

2.1 Introduzione

Con l'evoluzione della rete internet e l'introduzione del sistema digitale in campo televisivo è stato possibile introdurre nuovi servizi per gli utenti nell'ambito dell'intrattenimento. Il Video on Demand nasce infatti come servizio interattivo basato su queste due tecnologie. Da una parte nasce l'IPTV [16], la televisione attraverso internet, che permette a chiunque di accedere a contenuti multimediali attaverso la rete globale. Questo può avvenire semplicemente connettendosi al sito internet di un qualsiasi fornitore di servizi multimediali e guardando sullo schermo del proprio computer tutti i contenuti che vengono messi a disposizione. In alternativa è possibile anche utilizzare dei decoder che tramite la rete internet, scaricano contenuti multimediali che poi sarà possibile visionare sull'apparecchio televisivo collegato. Lo sviluppo dell'IPTV ha avuto luogo grazie soprattutto all'introduzione e ai continui miglioramenti della banda larga che ha reso possibile il trasferimento di grandi file in maniera rapida. La visione di contenuti tramite IPTV può avvenire secondo tre modalità:

streaming la visione avviene in contemporanea con il download

download streaming la visione avviene successivamente al download di una parte del contenuto video

download la visione avviene soltanto dopo il completamente del download del contenuto

L'IPTV, soprattutto negli ultimi anni, è stato introdotto in ambito televisivo il sistema di trasmissione digitale. Quest'evoluzione ha permesso un incremento nel flusso di informazioni trasmesse tramite i vecchi canali di trasmissione analogiche ed ha reso possibile, anche in questo ambito, la creazione di servizi VoD accessibili via televisione semplicemente tramite l'uso di un decoder digitale. Mentre l'IPTV è un servizio già presente da qualche anno e già completamente sviluppato, il VoD attraverso il sistema digitale è ancora in via di sviluppo e solamente da poco alcuni provider hanno reso possibile contenuti multimediali che l'utente può richiedere a piacere.

L'evoluzione continua del VoD sta portando ad un incremento esponenziale delle informazioni che vengono rese disponibili agli utenti. Per questo si è resa necessaria l'introduzione di un meccanismo che possa, in qualche modo, trasmettere all'utente soltanto le informazioni corrette e adatte alle sue esigenze. Le due tecniche principali per la messa in atto di questo meccanismo sono l'*Information Retrieval* (IR) e l'*Information Filtering* (IF). L'IR (lett. recupero di informazioni) consiste nel recupero, nella memorizzazione e nell'organizzazione di informazioni e ne sono esempio principale i nel campo del VoD la ricerca nel catagolo effettuata con il telecomando o tramite parole chiave. Attraverso l'inserimento di parole chiave, un sistema di IR permette all'utente di accedere in modo mirato alle informazioni di suo interesse. Questi risultati dipendono fortemente dalle parole chiave inserite, senza considerare chi ha richiesto tale servizio, ragion per cui non sempre posson risultare di interesse per l'utente stesso. La ricerca via telecomando invece ha lo svantaggio di essere noiosa per l'utente: scorrere tra le centinaia di film disponibili a catalogo può infatti essere un'operazione lunga e raramente effettuata dagli utenti. Un'evoluzione di questa tecnica, derivante dal fatto che l'IR può risultare impreciso o non sempre corretto è l'IF (lett. filtraggio di informazioni). La principale differenza dall'IR è che in questo caso viene tenuto in considerazione il soggetto che effettua la ricerca, grazie a questo l'IF è in grado di aiutare maggiormente l'utente nella ricerca di informazioni personalizzate. Questa tecnica sta alla base dei sistemi di raccomandazione (*Recommender Systems*), principale argomento di questo lavoro. Questi sistemi, nati a partire dagli anni '90, si sono ritagliati pian piano uno spazio sempre più importante nell'ambito della ricerca universitaria e soprattutto industriale con particolare attenzione all'area di e-commerce. Questo perchè, da un lato, viene data la possibilità all'utente di scovare in maniera più semplice prodotti di proprio gradimento, mentre dall'altro, i sistemi di e-commerce (ma non solo) hanno la possibilità di incrementare notevolemente le vendite andando a proporre agli utenti prodotti che saranno probabilmente di maggior interesse per loro stessi.

E' importante sottolineare il fatto che un sistema di raccomandazione ha come input principale il profilo dell'utente, cioè la storia di quest'ultimo all'interno del sistema, e in base questo va a predire nuovi prodotti nella speranza che questi risultino di suo gradimento. Questi oggetti vengono comunemente chiamati *item*. Ogni singolo item è descritto da un insieme di informazioni (nel caso di un film, ad esempio, il titolo, la trama, il cast). Ogni singola informazione verrà chiamata in seguito *metadato*. Le nuove raccomandazioni vengono elaborate sulla base dei gusti passati dell'utente, proprio perchè queste solo le poche e uniche informazioni disponibili, che un sistema di raccomandazione può avere sugli utenti; raramente infatti quando un utente si iscrive ad un fornitore di servizi come questo, rilascia tutte le sue informazioni personali intese non solo come anagrafica ma anche come gusti, hobby e tutte quelle informazioni che servono a caratterizzarlo. Tramite i sistemi di raccomandazione è inoltre possibile dare più visibilità agli item considerati impopolari, sconosciuti dalla maggior parte degli utenti, i quali possono venire raccomandati dal sistema in maniera automatica se considerati adatti alle preferenze dell'utente. I sistemi di raccomandazioni si suddividono in due categorie principali: Content-Based e collaborative. Nel corso di questo di lavoro verranno presentate ampiamente entrambe le categorie.

2.2 Sistemi di Raccomandazione

I sistemi di raccomandazione sono uno strumento tecnologico di supporto alle aziende che forniscono prodotti quali contenuti video (Video on Demand, per esempio la TV di Fastweb)o letterari (es Amazon).In questo contesto i prodotti sono generalmente definiti *item*. L'idea di base è quella di andare a capire i gusti di un utente, in modo da poter consigliare altri possibili acquisti andando a creare per ogni singolo utente una sorta di mercato personalizzato. Nel corso di questa tesi verranno presi in considerazione i sistemi di raccomandazione utilizzati nell'ambito del Video on Demand, per cui verranno trattati come utenti coloro che richiedono contenuti video e come item i film del catalogo VoD. Un sistema di raccomandazione necessita sostanzialmente delle informazioni sulle interazioni tra gli utenti e gli item, ossia un insieme di informazioni relative al fatto che un generico utente sia entrato in contatto con un determinato film. Questa interazione consiste semplicemente o in un voto che l'utente esprime esplicitamente o tramite una sorta di flag che indica che l'utente i ha interagito con il film j. Questo storico utente viene chiamato *profilo utente*. In aggiunta a queste informazioni possono essere disponibili i seguenti dati:

- Dati relativi ai film che vengono messi a disposizione del sistema (in questo caso per esempio genere, cast o trama) che rappresentano la gran parte dei dati disponibili al sistema, questo grazie al fatto che è nell'interesse dell'azienda fornire il maggior numero possibile di informazioni sui propri prodotti.

- Dati relativi agli utenti che utilizzano i servizi (per esempio gli utenti registrati ad un sito di e-commerce o gli abbonati alla IPTV); generalmente non è sempre possibile reperire un numero sufficiente di informazioni su di essi sia per motivazioni legate alla privacy, sia per la scarsa attitudine degli utenti nel fornire informazioni personali (spesso questi dati vengono raccolti attraverso questionari mirati la cui compilazione può risultare noiosa).

Questo insieme di informazioni, a seconda delle varie richieste da parte dell'utente, viene elaborato da un algoritmo di raccomandazione che permette di generare gli output del sistema di raccomandazione che possono essere di due tipi [27]:

Individual scoring : consiste in una predizione di un valore numerico che rappresenta l'eventuale rating che darebbe l'utente a quel film

Top-N recommendation : consiste in una lista di film, precisamente N film, che vengono consigliati all'utente e che dovrebbero risultare di suo gradimento

E' importante soffermarsi sulla rappresentazione delle informazioni disponibili in un sistema di raccomandazione e sull'architettura che sta alla base di questi ultimi. Entrambi gli argomenti saranno trattati nei paragrafi successivi.

2.3 Dataset

Un *dataset* è composto da due elementi essenziali:

- La matrice ICM (Item Content Matrix, Figura 2.1) che descrive le principali caratteristiche (metadati) di ogni film. Ogni singolo elemento I_{cj} di questa matrice rappresenta l'importanza della caratteristica c per l'item j. Tale matrice deriva dall'analisi del set di informazioni fornito dal content provider. Esempi di caratteristiche comprendono il titolo del film, gli attori, i registi etc. In un ambiente reale possiamo incontrare informazioni inaccurate perchè ogni giorno potremmo avere

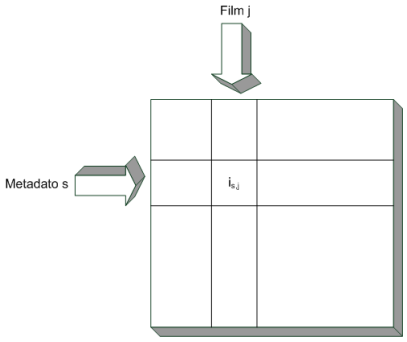


Figura 2.1: Matrice ICM

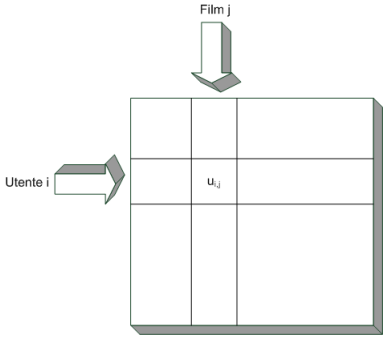


Figura 2.2: Matrice URM

aggiunta di contenuti. Questa matrice viene utilizzata negli algoritmi Content-Based (questo tipo di algoritmi verrà descritto nella Sezioni 2.5.1).[4]

- La matrice URM (User Rating Matrix, Figura 2.2), ovvero l'insieme degli storici di tutti gli utenti iscritti al sistema. Supponendo che nel sistema sono presenti m utenti e n item, la URM avrà dimensioni $m \times n$ in cui ogni riga i corrisponde ad un diverso profilo utente e ogni colonna j corrisponde ad un film che il sistema mette a disposizione degli utenti. Se indichiamo con R la matrice URM, ogni singolo elemento di essa r_{ij} corrisponderà sostanzialmente alla votazione (preferenza) fornita da parte dell'utente i per l'item j. Questi voti sono fondemaentali in quanto è in base ad essi che viene effettuata la raccomandazione. Si considei che una classe di algoritmi chiamati Collaborativi (vedi Sezione 2.5.2) usano solo queste informazioni per generare la raccomandazione. Facendo riferimento all' URM, possiamo dividere i vari dataset in due categorie in base alla modalità con la quale i rating sono stati considerati:

- Rating Impliciti
- Rating Espliciti

Nei paragrafi seguenti vengono presentate le differenze principali tra le due categorie.

2.3.1 Rating Espliciti

Nei dataset che contengono rating espliciti viene tenuta in considerazione la preferenza dell'utente riguardo ad un determinato film, anche se questi rating possono affetti da distorsioni [3]. I rating vengono assegnati all'interno di una scala di valori (ad esempio da 1 a 5, oppure da -2 a +2), dove lo 0 rappresenta la non interazione tra l'utente e il film, i voti più bassi vengono considerati negativi mentre i più altri positivi. Un'eventuale raccomandazione basata su un dataset di questo tipo sarà sicuramente più accurata (precisa) di una basata su dei rating impliciti. Nonostante ciò emergono nuove problematiche:

- Le votazioni esplicite sono soggette alla personale interpretazione della votazione stessa da parte dell'utente: ad esempio in una scala di voti che va da 1 a 5, una voto pari a 3 per un utente potrebbe corrispondere a un voto positivo da parte di un utente o a un brutto voto da parte di un altro [4]
- Alcuni algoritmi di raccomandazione effettuano la raccomandazione andando ad ordinare i film in base al voto predetto per quel determinato film. A seconda dell'algoritmo questo voto può variare in un determinato range. Range che spesso non è uguale al range di voti di un dataset esplicito. E' quindi necessaria una sorta di conversione tra i vari range.

2.3.2 Rating Impliciti

Nei dataset contenenti rating impliciti si tiene conto esclusivamente se un utente ha visto un determinato film oppure no. Di conseguenza riferendosi di nuovo alla matrice URM, ogni singolo elemento r_{ij} potrà essere 0 (l'utente non ha visto il film) oppure 1 (l'utente ha interagito con il film). Sostanzialmente è il sistema ad inferire voti basandosi su un'analisi del comportamento dell'utente, essendo nel campo del VoD possiamo considerare $r_{ij} = 1$ come l'utente i ha acquistato il film j.Tramite l'utilizzo di questi rating viene quindi trascurata l'eventuale preferenza dell'utente in merito ad un film. Un dataset di questo tipo può sicuramente portare ad una raccomandazione

meno accurata a causa del fatto che tutti i film visti da un singolo utente sono messi sullo stesso piano. Si pensi, ad esempio, ad un profilo utente composto da due voti impliciti in cui un film è stato gradito dall'utente, mentre l'altro no, il sistema non potrà interpretare questa differenza di gradimento e potrebbe dar maggior peso al secondo film per la costruzione di raccomandazioni future. La necessità di avere dei rating impliciti è portata dal fatto che non sempre un sistema di raccomandazione riesce a ricevere dagli utenti delle preferenze sui film con cui hanno interagito.

2.3.3 Evoluzione temporale di un dataset

Un sistema di raccomandazione è in continua evoluzione nel tempo. Essendo un sistema che mantiene informazioni relative alle interazioni con gli utenti, i cambiamenti dei dati sui quali sono basate le raccomandazioni sono continui, specialmente in sistemi che gestiscono molti utenti e molti item. Per avere le idee più chiare si pensi semplicemente ad un migliaio di utenti che vedono uno o più film, in un lasso di tempo t: i loro profili utenti, dopo la visione, verranno modificati e ciò influirà sulle successive raccomandazioni. Possiamo individuare tre tipi di cambiamenti all'interno di un dataset:

- L'introduzione di un nuovo voto in un profilo utente, ossia quando un utente interagisce con un film e esprime un rating (dataset esplicito) oppure viene inserita l'interazione utente-item all'interno del dataset (dataset implicito).

- L'introduzione di un nuovo film nel sistema, ossia quando viene messo a disposizione degli utenti un nuovo film che, sostanzialmente, significa aggiungere una colonna all'URM che tenga conto delle interazioni del nuovo item con gli utenti.

- La registrazione di un nuovo utente al sistema che comporta, in pratica, l'aggiunta di una nuova riga all'interno della URM che rappresenti il profilo del nuovo utente.

E' importante sottolineare il fatto che, in un sistema di raccomandazione reale, questi cambiamenti avvengono continuamente ed è necessario far fronte ad essi in modo da mantenere un certo livello di accuratezza nelle raccomandazioni. La raccomandazione, infatti, si basa su un modello, creato a partire dall'URM ad un determinato tempo t. Questo modello, poi, viene aggiornato in tempi successivi a discrezione del gestore del sistema di raccomandazione. Se un modello viene aggiornato dopo un lasso di tempo elevato, le

raccomandazioni prima di tale aggiornamento potrebbero non essere accurate. Al contrario se il modello venisse aggiornato dopo un lasso di tempo breve, le raccomandazioni risulterebbero molto più accurate ma questo aggiornamento comporterebbe un carico computazionale molto elevato. Per capire meglio la mole di lavoro che può causare una serie di cambiamenti di questo tipo, verrà introdotta nel paragrafo successivo una breve descrizione dell'architettura di un sistema di raccomandazione.

2.4 Architettura di un sistema di raccomandazione

L'architettura di un sistema di raccomandazione è articolata in due fasi: una fase Batch e una fase Real-Time. Proprio per rispettare i requisiti real-time (si ricorda che un sistema di raccomandazione deve essere sempre pronto a raccomandare qualcosa agli utenti), tale sistema e gli algoritmi su cui è basato seguono un approccio basato su modelli.
La fase di batch crea una rappresentazione (modello) dei dati ricevuti in input dal sistema. Tale fase spesso richiede elevate risorse computazionali ed è per questo che viene eseguita durante le ore in cui il sistema non è in sovraccarico. E' importante però che questo aggiornamento avvenga a intervalli regolari, in modo da non utilizzare modelli troppo vecchi per le raccomandazioni. Tali intervalli devono tener conto della frequenza di aggiunta di item, utenti e voti all'interno del sistema stesso. In ogni caso, dataset realistici composti da milioni di utenti e di film possono avere dei requisiti computazionali veramente pesanti. Per fortuna le matrici come l'URM e l'ICM sono molto sparse: spesso molti utenti interagiscono solo con una piccola parte dei film messi a disposizione. Il vantaggio è che matrici sparse possono essere utilizzate attraverso rappresentazioni molto efficienti e poco costose.
La parte real-time, invece, utilizza il modello creato dalla fase precedente per soddisfare le richieste degli utenti a real-time. E' importante che tali risposte pervengano all'utente nel minor tempo possibile, altrimenti l'utente potrebbe perdere fiducia nel sistema stesso[4]. Quest'architettura deve permettere inoltre che la raccomandazione possa essere effettuata sia per degli utenti generici con profili aggiornati, e quindi non con il profilo utente presente nel momento della creazione del modello sia per degli utenti nuovi non contenuti nella URM nel momento in cui il sistema genera il modello. A causa di questo però viene vincolato il tipo algoritmo utilizzabili. Non è possibile infatti utilizzare degli algoritmi che modellizzino esplicitamente l'utente, ma che siano in grado di vedere ogni utente, sia pre-esistente

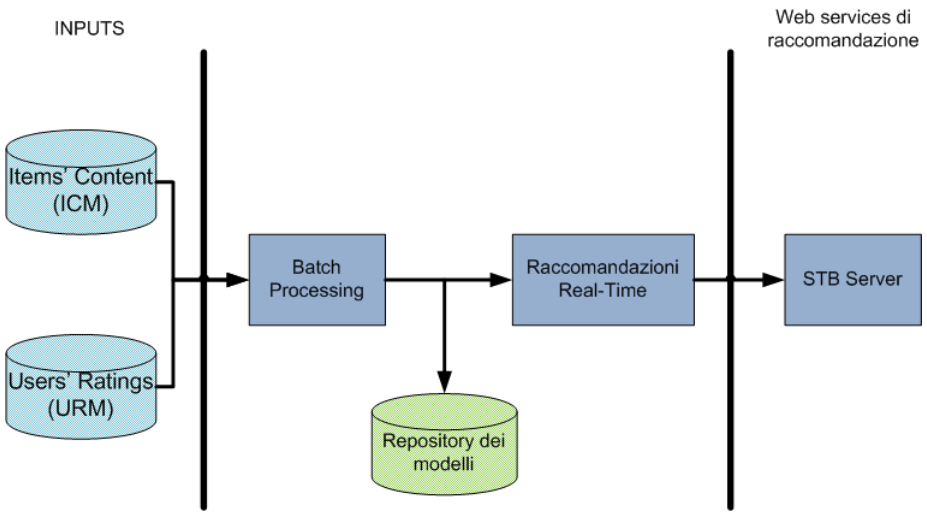


Figura 2.3: Fasi di batch e real-time

che nuovo, come funzione degli item con cui ha interagito, formalmente utente = f(item con rating).

2.5 Algoritmi di raccomandazione

Una raccomandazione dipende da diversi fattori: il profilo dell'utente che l'ha richiesta, il modello su cui si basa la raccomandazione stessa, la popolarità dei film etc. Ma ciò che sta alla base di un sistema di raccomandazione e ciò che utilizza questo insieme di informazioni non è nient'altro che un algoritmo, più o meno complesso, di raccomandazione. A grandi linee, un algoritmo riceve in input la URM aggiornata ad un determinato tempo t e attraverso essa costruisce un modello. Quando poi dovrà essere effettuata una determinata raccomandazione per un profilo utente u, l'algoritmo utilizzerà il modello creato e il profilo utente per costruire la lista di raccomandazione. Ogni algoritmo di raccomandazione, ovviamente, ha le sue peculiarità e può essere più o meno accurato (analizzeremo poi in seguito le metriche per studiare la bontà di un determinato algoritmo). Nei seguenti paragrafi verranno spiegate le varie categorie in cui si possono suddividere gli algoritmi di raccomandazione e gli algoritmi utilizzati. Nella Figura 2.4 viene presentata una classificazione degli algoritmi esistenti nella quale sono evidenziate le tipologie di algoritmo conformi all'architettura presentata nella Sezione 2.4 [27].

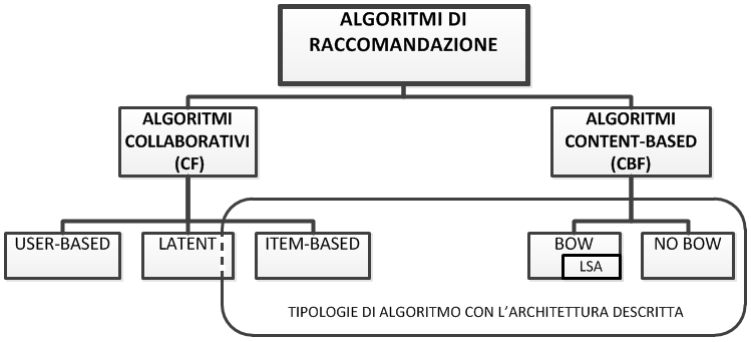


Figura 2.4: Classificazione algoritmi di raccomandazione

2.5.1 Algoritmi Content-Based

Un sistema Content-Based è basato sull'analisi del contenuto dei film proposti dal sistema stesso agli utenti [23]. Sostanzialmente, in un sistema del genere, all'utente verranno raccomandati film molto simili a quelli di cui l'utente ha fruito. In base all'analisi di contenuto sul profilo utente, verrà effettuata la raccomandazione. In un primo momento questa analisi dei contenuti potrebbe risultare molto conveniente e efficiente, proprio perchè rappresenta forse la soluzione più semplice in assoluto (in base a quello che l'utente vede, il sistema raccomanda qualcosa che, ad esempio, appartiene allo stesso genere o ha un determinato attore etc. In realtà è proprio questa analisi dei contenuti che può portare a delle problematiche: spesso i vari film hanno molte cose in comune (alcuni membri del cast, stesso regista, stesso genere) e soprattutto alcune di queste informazioni possono introdurre qualche ambiguità (si pensi ad esempio al fatto che un film possa appartenere a più generi diversi) nell'ambito della raccomandazione. Si può comunque scegliere su quali informazioni basare la raccomandazione stessa, andando a limitare la quantità di dati da dare in pasto all'algoritmo. L'utilizzo di questo approccio dipendente dal contenuto permette di individuare quali interessi portano un individuo a scegliere un determinato item. Vengono usati diversi metodi per determinare la similarità tra il profilo degli item e il profilo degli utenti, come ad esempio il Coefficiente di Correlazione di Pearson. Tali algoritmi hanno il vantaggio di poter raccomandare item che non sono mai stati fruiti, proprio perchè si basano sulle caratteristiche degli item per generare la raccomandazione. Ciononostante essi soffrono però di alcune limitazioni [1] :

- La raccolta delle informazioni sugli item richiede che esse siano in un formato automaticamente gestibile dal calcolatore.

- L'estrazione di caratteristiche significative è un problema da non sottovalutare.Tale problema deriva sia dal formato dei dati sia dalla sparsità di informazioni relative alle caratteristiche degli item. Capita a volte che di qualche film si conoscano poche informazioni oppure molte informazioni (tra cui ad esempio la trama etc.).

- Nel caso due film siano rappresentati dalle stesse caratteristiche essi risulteranno, agli occhi dell'algoritmo, sostanzialmente identici. Di conseguenza l'algoritmo stesso non può distinguere un film gradevole per l'utente da uno sgradevole se entrambi, ad esempio, hanno lo stesso regista o buona parte delle stesso cast di attori [4].

- Le raccomandazioni sono limitate ad item simili a quelli già fruiti dall'utente. Un item con caratteristiche mai fruite dall'utente non verrà mai raccomandato. Se un utente non ha mai visto un film horror è probabile che il sistema non gliene raccomanderà mai uno,anche se magari all'utente potrebbe piacere.

- Non è possibile generare una raccomandazione per un utente nuovo, che non abbia ancora votato alcun item. Inoltre maggior è il numero di item votati, migliore sarà la raccomandazione. Di conseguenza un utente con un profilo poco ricco di rating non riceverà una buona raccomandazione.

Algoritmo Content-Based diretto

L'algoritmo Content-Based di tipo diretto si basa sulle meta-informazioni relative ai film ed è composto dalle seguenti fasi:

1. Tokenizzazione
2. Rimozione delle Stop Words
3. Stemmizzazione
4. Assegnazione di un peso ai metadati (TF-IDF)

Ricevendo come input le meta-informazioni relative ai film, viene effettuata inizialemente un'operazione detta *tokenizzazione* che consiste nella suddivisione in singole parole, detti token, delle stringhe contenute all'interno delle meta-informazioni. Quest'operazione viene effettuata per cercare di individuare un numero frequente di ripetizioni di alcune parole chiave. Si tenga presente che la suddivisione non è sempre netta, ma in alcuni casi,

come ad esempio nomi e cognomi di attori, un token sarà formato da più parole.

L'output di tale operazione viene poi filtrato basandoci su un certo dizionario contenente una serie di parole, dette *stop words*, che potrebbero risultare ininfluenti nelle fasi successive (ad esempio articoli, congiunzioni, preposizioni, etc.), se un token corrisponde ad una di queste parole verrà scartato.

La fase seguente, che riceve in input la serie di token filtrati, viene chiamata *stemmizzazione*. Questa fase consiste nel modificare alcuni token, tranne ad esempio i nomi propri o i nomi di luoghi, togliendo il suffisso (a seconda della struttura della parola). Le parole risultanti vengono dette stem e vengono generate in modo che parole simili (con lo stesso prefisso) siano rappresentate dallo stesso stem.

Esempio : i token uccidere, uccide, ucciso,uccisione possono essere riconosciuti dallo stesso stem ucci.

Il risultato dell'operazione di stemming consiste in un insieme di stem e di token (quelli che non son stati modificati durante la fase di stemming) detto BOW (Bag Of Words) [5].

A questo insieme di parole viene successivamente effettuata un'analisi statistica per il calcolo del peso di ogni singola parola in merito alla sua presenza nelle meta-informazioni di ogni film. Viene calcolato il TF-IDF (Term Frequency-Inverse Document Frequency) [24]. Il Term Frequency è un peso che può consistere nel numero delle occorrenze di una certa parola nell'ambito di uno specifico oggetto. L'Inverse Document Frequency è un peso calcolato come l'inverso del numero di occorrenze di una certa parola in tutto il catalogo di oggetti (vi sono diversi metodi di calcolo che portano a diversi vantaggi e svantaggi). Ad entrambi i pesi si può applicare una normalizzazione dividendo il valore per il numero totale di parole contenute nel BOW. Da questi due pesi ne viene calcolato uno unico che viene utilizzato per la costruzione della matrice ICM (Item Content Matrix). In tale matrice, che costituisce il vero e proprio modello prodotto dall'algoritmo, le righe corrispondono alle singole parole (stem o token) mentre le colonne corrispondono ai film. Ogni singola cella contiene il valore del peso calcolato attraverso TF-IDF.

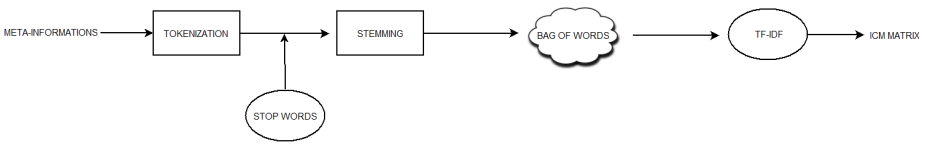


Figura 2.5: Procedimento algoritmo Content-Based diretto

Le colonne della matrice rappresentanti gli item, attraverso una serie di stem/token che li caratterizzano, formano dei vettori che vengono disegnati su un piano i cui assi sono formati dagli stessi stem. Successivamente si disegna nel piano il vettore utente, calcolato come la sommatoria dei voti espressi dall'utente moltiplicati per la colonna (della matrice ICM) relativa al film a cui corrisponde il voto.

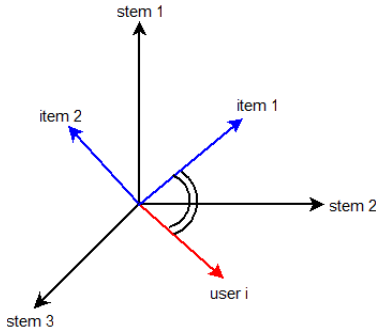


Figura 2.6: Vettori Utente e Item

Il fatto che un item possa essere consigliato ad un utente viene stabilito considerano l'ampiezza dell'angolo tra il vettore utente e il singolo item. In particolare ne viene generalemente calcolato il coseno in modo da avere un risultato compreso tra un range ridotto di valori, -1 e 1, o addirittura 0 e 1 nel caso di voti impliciti. Gli N vettori rappresentanti i film che avranno i valori del coseno più alti (che indicano quindi similarità) verranno scelti ed inseriti nella lista Top-N.

2.5.2 Algoritmi Collaborativi

Questo tipo di algoritmi cerca di suggerire item ad un particolare utente sulla base delle preferenze espresse dagli altri utenti del sistema [11]. L'approccio collaborativo, in qualche modo, simula il passaparola umano, ad esempio quando un utente non sa quale film noleggiare chiede qualche consiglio ad amici con gusti simili o cerca su Internet qualche recensione [26]. Questi algoritmi, quindi, si basano sull'opinione di utenti della community per trovare individualmente per ogni utente dei contenuti pertinenti e d'interesse per l'utente stesso. Si basano sostanzialmente su due assunzioni [4]:

- Gruppi di utenti con gusti simili, votano gli item in maniera simile. Questo viene chiamato concetto di vicinanza tra utenti.
- Gli item correlati tra loro vengono votati dagli utenti in maniera simile. Questo viene chiamato concetto di vicinanza tra item.

Quest'ultima assunzione spiega, sostanzialmente, che un determinato item è simile ad un altro, non tanto per i contenuti in comune, ma semplicemente perchè viene votato in maniera simile da parte di un set di utenti all'interno della community. Proprio per questo motivo i due item sono vicini tra loro. Gli algoritmi Collaborativi a differenza di quelli Content-Based possono portare al cosiddetto effetto serendipity (effetto sorpresa) [13]. Infatti in una lista costruita tramite un algoritmo Content-Based, l'utente sarà sempre in grado di capire il motivo per cui un certo oggetto gli è stato consigliato (per esempio nel caso di film, perché questo è del suo genere preferito, perché è presente il suo attore preferito, etc.). Nelle liste costruite tramite algoritmi Collaborativi, invece, potranno essere presenti raccomandazioni che potrebbero stupire l'utente, in quanto non derivanti da ciò con cui lui è entrato in contatto. Rispetto agli algoritmi Content-Based l'approccio Collaborativo ha il vantaggio di essere indipendente dal contenuto, e di non necessitare quindi di altri tipi di informazioni su utenti (per esempio età, sesso, etc.) e item (per esempio nel caso gli item siano film avremo titolo, genere, etc.). Essi presentano però anche dei limiti [1]:

- Problema dell'utente nuovo: un utente che non ha ancora espresso un voto su alcun item non può essere in qualche modo associato a nessun altro utente.
- Problema dell'item nuovo: se un item non è mai stato votato allora non può comparire nella lista di raccomandazione, proprio perchè non risulta simile a nessun altro item visto dall'utente.

- La raccomandazione è legata alla sparsità della matrice URM: più una URM è ricca di informazioni, più accurata sarà la raccomandazione.

- Utenti con gusti particolari saranno difficilmente associabili ad altri utenti e la raccomandazione risulterà meno accurata.

All'interno della categoria degli Algoritmi Collaborativi possiamo effettuare un'ulteriore suddivisione [28]:

- User-Based: si concentrano sul concetto di utenti simili. L'idea di fondo è quella di stabilire una relazione di similarità tra gli utenti della comunità in modo da poter costruire un modello che consista in una matrice $m \times m$ (dove m è il numero di utenti della comunità, o il numero di utenti esaminati) in cui sia le righe che le colonne rappresentano un utente del sistema. Questo tipo di algoritmi può avere problemi a livello di performance durante la costruzione del modello a causa dell'eccessiva mole di dati che si ha a disposizione (si pensi che una comunità IPTV, ad esempio, può avere decine di migliaia di utenti). Un altro tipico problema è relativo all'inserimento di un nuovo utente all'interno della comunità: tale operazione, infatti, richiederebbe la completa ricostruzione dell'intero modello (ulteriore carico computazionale).

- Item-Based: si concentrano sul concetto di oggetti simili. L'idea di fondo è quella di stabilire una relazione di similarità tra gli oggetti forniti dal servizio in modo da poter costruire un modello che consista in una matrice $n \times n$ (dove n è il numero di oggetti forniti dal servizio, tale matrice verrà indicata come D) in cui sia le righe che le colonne rappresentano un oggetto. E' opportuno introdurre anche come tali algoritmi riescano a produrre una lista di raccomandazione. L'input del sistema di raccomandazione, come già abbiamo detto, è la User Ranking Matrix R. Il primo passo consiste nella costruzione del modello, ossia della matrice D, a partire dalle informazioni presenti all'interno della matrice R. Una volta costruito il modello, il sistema è pronto per essere interrogato. Se ad un certo punto è necessario consigliare nuovi oggetti ad un utente, semplicemente si moltiplica il vettore relativo all'utente stesso per la matrice D. Questa operazione da come risultato una lista di raccomandazione che, successivamente, potrà essere modificata.

Algoritmo collaborativo diretto

L'algoritmo collaborativo diretto, come tutti gli algoritmi Collaborativi, si basa sulle opinioni della comunità di utenti. Questo particolare tipo di algoritmo per definire la similarità tra due film presuppone che se un utente ha guardato due film, questi due film si possono considerare simili. Il modello che va a costruire consiste in una matrice D di similarità in cui sia le righe che le colonne rappresentano i film. Il singolo elemento $d_{i,j}$ indica il grado di similarità tra i due film calcolato come il numero di occorrenze nelle righe della matrice URM dove sono presenti sia il film i che il film j. Naturalmente gli elementi sulla diagonale sono tutti 1. Questo valore, a volte, può subire una sorta di normalizzazione del coseno attraverso la seguente formula [10]:

$$d_{i,j} = \frac{\#\text{Occorrenze}(i,j)}{(v(\#\text{Occorrenze}(i)) \cdot v(\#\text{Occorrenze}(j)))} \tag{2.1}$$

la quale mi da un risultato compreso tra -1 e 1, o nel caso di valutazioni implicite tra 0 e 1 (aggiorna con parte item popolari etc). Moltiplicando il vettore utente binario(la riga della URM di interazione utente-item) per la matrice D ed ordinando i risultati ottengo un vettore i cui primi N elementi sono gli N film da consigliare. Una variante di questo algoritmo viene detta KNN e consiste nell'analizzare la matrice D, facendo una sorta di filtraggio degli elementi eliminando un gran numero di elementi fino a tenerne soltanto K. Dalla matrice D vengono infatti eliminati, cioè settati a 0, per ogni riga/colonna i valori più bassi. Tali valori, infatti, sono ininfluenti nella produzione della lista di raccomandazione finale perché potrebbero da un lato portare a delle variazioni pressoché nulle della lista, da un altro potrebbero agire come una sorta di rumore e quindi non garantire una sensata raccomandazione. I risultato di tale operazione porta ad ottenere una matrice D sparsa (di più facile memorizzazione ed elaborazione) con un numero ridotto di informazioni, ma considerate tutte utili, e quindi con una quasi inesistente parte di rumore.

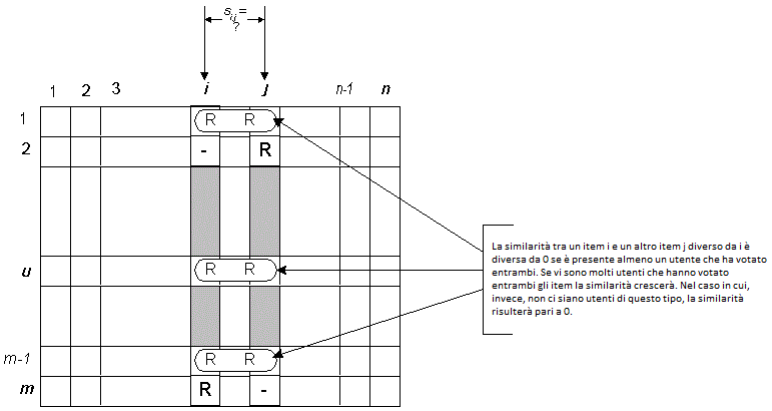


Figura 2.7: Calcolo della similarità tra item

Ulteriori considerazioni sugli algoritmi di tipo collaborativo

Nelle due sottocategorie di algoritmi Collaborativi, il modello viene costruito in modo simile, ma speculare. In entrambi i casi si cercano le similarità, tra gli utenti negli User-Based, tra gli oggetti negli Item-Based. Il punto di partenza è sempre la matrice URM, in un caso (User-Based) vengono prese le righe della matrice, mentre nell'altro (Item-Based) le colonne. Abbiamo quindi dei vettori di utenti o di oggetti che costituiranno delle coordinate in un piano.

- User-Based: i vettori riga della matrice URM (utenti) vengono disegnati in un piano i cui assi rappresentano gli oggetti. Per individuare le similarità tra gli utenti quindi vengono analizzati gli angoli tra i vettori-utente. Un angolo poco ampio individua una maggior similarità tra due utenti, e quindi un possibile 1 nella matrice del modello Utente-Utente. Un angolo ampio indica una netta differenza tra due utenti che verrà rappresentata con uno 0 nella matrice. Nell'esempio seguente (Figura 2.8) si possono vedere gli utenti $u1$ e $u3$ che appartenendo allo stesso piano vengono considerati simili. Naturalmente la matrice risultante avrà tutti 1 sulla diagonale (Figura 2.9).

- Item-Based: specularmente agli Item-Based utilizzo i vettori colonna rappresentatni gli item (Figura 2.10) disegnandoli su un piano dove gli assi rappresentano gli utenti. Tramite lo studio degli angoli andremo a costruire una matrice di similitudine Item-Item (Figura 2.11).

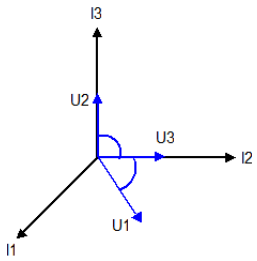


Figura 2.8: Esempio vettori utente

	U1	U2	U3
U1	1	0	1
U2	0	1	0
U3	1	0	1

Figura 2.9: Matrice ricavata dai vettori utente

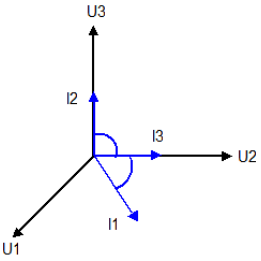


Figura 2.10: Esempio vettori oggetti

	I1	I2	I3
I1	1	0	1
I2	0	1	0
I3	1	0	1

Figura 2.11: Matrice ricavata dai vettori oggetti

In termini di prestazioni la costruzione del modello negli algoritmi Item-Based impiegherà un tempo minore rispetto agli User-Based dato che in un qualsiasi sistema il numero di utenti è generalmente di gran lunga superiore al numero dei item presenti. Questo comporta un numero inferiore di vettori da analizzare nella ricerca delle similitudini. Un altro aspetto da valutare in entrambi i casi sono le conseguenze in termini di aggiornamento del modello causate dall'aggiunta o dalla modifica di un utente o di un oggetto al sistema. Consideriamo innanzitutto che la frequenza con cui vengono aggiunti nuovi utenti al sistema è superiore alla frequenza con cui vengono aggiunti degli oggetti. Nel caso di User-Based il modello andrà ricostruito ogni volta che viene aggiunto un sistema, mentre nel caso di Item-Based in caso di nuovi oggetti. A questo bisogna poi aggiungere il fatto che una riga della matrice R sarà meno popolata di una colonna, infatti si può ipotizzare che i voti ricevuti da un oggetto(colonna) sono maggiori rispetto a quelli espressi da un utente(riga). Da queste considerazioni possiamo osservare che :

- Il modello di un algoritmo User-Based andrà rifatto più volte di uno Item-Based

- L'aggiunta di nuovi utenti porta ad una ricostruzione del modello nel caso di User-Based, infatti un nuovo utente comporta un aumento delle dimensioni del modello utilizzato (una nuova riga e una nuova colonna nella matrice utente-utente). Considerando il possibile numero di utenti di una comunità e la frequenza di aggiunta degli utenti questa ricostruzione potrebbe portare ad un eccessivo carico computazionale. Per quanto riguarda gli Item-Based, il modello rimane identico (la matrice utilizzata come modello considera le similitudini tra item).

- L'aggiunta di nuovi oggetti porta ad una ricostruzione del modello utilizzato dagli Item-Based (caso speculare al precedente degli User-Based). Negli algoritmi Collaborativi in generale, l'inserimento di un nuovo oggetto è un fattore molto delicato: se viene aggiunto un nuovo item al catalogo, infatti, non saremo mai in grado di consigliarlo fintanto che non riceverà un numero sufficiente di voti (ricordiamo ancora una volta che i Collaborativi si basano sui voti espressi dalla comunità di utenti). Di conseguenza il fatto di non aver un modello item-item aggiornato non ha un grosso impatto sul sistema.

- Oltre all'aggiunta di utenti e item, la ricostruzione del modello è necessaria anche nel caso di modifiche degli stessi. Sulla base della considerazione espressa in merito al diverso numero di voti a disposizione

per utenti e item, una modifica del profilo utente (l'aggiunta o la cancellazione di un voto) avrà un impatto maggiore rispetto all'aggiunta (o alla cancellazione) di un nuovo voto per un oggetto. Di conseguenza con un Item-Based siamo in grado di catturare dinamicamente le modifiche di un profilo utente, senza, ogni volta, rigenerare il modello.

2.6 Lavori Correlati

L'argomento trattato in questo lavoro rientra nella macrocategoria di studio sull'evoluzione temporale di un sistema di raccomandazione. Questo studio, in particolare, analizza il comportamento di un sistema di raccomandazione al crescere del dataset, considerando la capacità di predizione nel tempo degli algoritmi di raccomandazione in base all'inserimento, nel dataset, di nuovi utenti o di nuovi film[8].

Uno dei problemi più discussi nell'ambito dell'evoluzione temporale di un sistema di raccomandazione è il Cold-Start[7], ovvero la mancanza di informazioni che si presenta con l'aggiunta, nel dataset, di nuovi utenti e di nuovi film, che rende molto difficile l'inferenza di nuove informazioni. Il problema si manifesta in maniera maggiore negli algoritmi di raccomandazione di tipo Collaborativo dove le predizioni di rating avvengono soltanto in base ai rating collezionati. Quando ci sono pochi rating i modelli collaborativi sono infatti inaccurati. L'unico lavoro che affronta un problema correlato a quello affrontato il questo lavoro di tesi, è il recente articolo di *Lathia et al.*[21] il cui argomento principale è lo studio dell'evoluzione temporale relativo al dataset completo del Netflix Prize. Nello studio trattato sono state effettuate delle misurazioni sulla Diversity (quella che in questo lavoro è chiamata dinamicità o Reshuffling) e sulla Novelty (vedi Capitolo 7) tramite le liste di raccomandazione prodotte in determinati istanti di tempo. Nell'articolo sono confrontati tra loro tre algoritmi in grado di fare delle predizioni sul voto di un certo item: baseline, che fa semplicemente la media dei rating di un item, Item-Item Cosine KNN e un algoritmo basato sulla decomposizione SVD.

L'esperimento effettuato su un dataset crescente nel tempo ha inizio ad un generico istante t. Viene fissata una finestra temporale μ che rappresenta gli istanti in cui verranno richieste le raccomandazioni al sistema. Ad ogni iterazione vengono calcolati, in base alle informazioni presenti al tempo $t + (n \cdot \mu), k \in \mathbb{N}$, i rating predetti per ogni item non votato precedentemente da ogni utente attivo nella finestra temporale, ovvero che ha inserito almeno un rating; vengono quindi prodotte una serie di liste Top-N, una per ogni utente attivo.

Il principale risultato mostra come la diversità delle liste di raccomandazione sia mediamente maggiore della Novelty, ciò significa che nel corso del tempo, se le liste di raccomandazione costruite per un certo utente cambiano, questo cambiamento porta a consigliare maggiormente dei film che sono già stati raccomandati in passato. Nell'articolo gli autori riportano le seguenti considerazioni che sono risultate utili in alcuni aspetti di questo lavoro:

- Esiste un legame tra la lunghezza del profilo utente e la presenza di Diversity.
- Un fattore importante per la presenza di Diversity è il numero di rating inseriti trai due istanti in cui vengono costruite (e confrontate) due liste di raccomandazione.
- Un altro fattore importante è l'intervallo di tempo che intercorre tra le due raccomandazioni di un determinato utente. In questo intervallo di tempo, infatti, altri utenti possono interagire con il sistema modificando implicitamente le relazioni trai diversi film e di conseguenza i nuovi modelli costruiti. Per questo più è lungo quest'intervallo di tempo più si rileva la presenza di Diversity.

In un certo senso, la tesi affronta un problema complementare a quello di *Lathia et al.*, mentre in [21] si cerca una metodologia per aumentare la dinamicità delle liste di raccomandazione, questo lavoro di tesi si occupa del problema opposto, ovvero cercare una metodologia per alleviare l'eccessiva dinamicità. Entrambe le situazioni sono fastidiose per l'utente.

A seconda del tipo di dominio i dataset e i relativi algoritmi di raccomandazione possono essere più o meno affetti da uno o dall'altro problema. Come verrà mostrato nel Capitolo 4, un tipico dataset IPTV (MovieRec) ha profili utente relativamente corti (cioè, con pochi rating noti per utente) e risulta perciò particolarmente affetto da eccessiva diversità.

Alcune considerazioni esposte in [21], vengono riprese in questo lavoro, in particolare nel Capitolo 4 oltre a considerare la lunghezza del profilo come possibile causa della presenza di diversità, viene introdotto anche il concetto di popolarità di un profilo o di un film, per comprendere come questa può influire sulla dinamicità, soprattutto in algoritmi Collaborativi.

Nel Capitolo 7 viene studiata l'evoluzione temporale di un dataset reale utilizzando due finestre temporali di lunghezza 7 giorni[21] e 28 giorni.

Capitolo 3

Algoritmi di raccomandazione utilizzati

Nei seguenti paragrafi verrano presentati gli algoritmi di raccomandazione utilizzati nel corso di questo lavoro.

3.1 Introduzione

In questo lavoro sono stati utilizzati degli algoritmi che funzionano secondo l'architettura descritta nella Sezione 2.4. Questi algoritmi sono sia Collaborativi che Content-Based, tra questi possiamo evidenziarne due che rappresentano lo stato dell'arte delle due categorie di algoritmi: l'AsymmetricSVD per la classe degli algoritmi Collaborativi e LSA Cosine per la classe dei Content-Based. In entrambe le categorie possiamo individuare alcuni algoritmi che non utilizzano la matrice URM, ma anche l'ICM, nelle loro dimensioni reali ma, tramite delle decomposizioni, lavorano in uno spazio a dimensioni ridotte. In questo capitolo il profilo utente verrà rappresentato tramite un vettore di lunghezza n, con n pari al numero di film presenti nel dataset. In ogni cella i è presente il rating, implicito e esplicito, dato dall'utente al film i, la mancanza di rating per un determinato film è rappresentata da uno 0.

3.2 Algoritmi con modello nello spazio latente

Alcuni algoritmi di raccomandazione lavorano in uno spazio ridotto, definito spazio latente. Questi algoritmi descrivono il dataset per mezzo di un set ridotto di *features* (caratteristiche nascoste di una matrice) ricavate at-

traverso una decomposizione. Queste *features*, il cui numero è variabile e viene indicato con l, possono rappresentare le seguenti cose:

- Se si utilizzano algoritmi Collaborativi, rappresentano implicitamente l'interazione tra gli utenti e gli item. In questo caso la decomposizione viene applicata alla matrice URM.
- Se si utilizzano algortimi Content-Based, rappresentano implicitamente il contenuto di ogni item, ovvero i metadati che descrivono ogni determinano item (che abbiamo definito come *stem*). In questo caso la decomposizione viene applicata alla matrice ICM.

Utilizzando un modello nello spazio latente è possibile rappresentare utenti e item in uno spazio più ridotto di quello originale, di dimensione $l << m, n, s$ con m e n dimensioni della matrice URM, s e n dimensioni della matrice ICM. Tramite l'utilizzo delle *features* è anche possibile ridurre il rumore perchè vengono eliminati i dati con meno contenuto informativo a beneficio di quelli più ricchi di informazioni[9]. Uno dei modi per ridurre le dimensione della matrice originale è la tecnica matematica denominata SVD (Singolar Value Decomposition)[15]. Questa tecnica prende in input una matrice M di dimensioni $a \times b$ e la dimensione latente l che rappresenta sia la dimensione dello spazio sia il numero di *features* che si va ad individuare in M. Il risultato di questa decomposizione è composto da tre matrici:

U matrice unitaria di dimensioni $a \times a$.

S matrice diagonale di dimensioni $l \times l$ che rappresenta attraverso dei valori sempre positivi le *features* (valori singolari della matrice M).

V' trasposta coniugata di una matrice unitaria di dimensioni $b \times b$.

Tramite questa decomposizione rafforzo la similarità tra gli item. Infatti se due item i e j rappresentati dai vettori $\vec{i}$ e $\vec{j}$ sono considerati simile nello spazio originale, una volta riportati nello spazio latente, i rispettivi vettori sono ancora più vicini. Infine la decomposizione SVD permette di trovare anche delle dipendenze nascoste tra gli item[2].

3.3 Algoritmi Item-Item Cosine KNN

La classe Item-Item Cosine è una classe di algoritmi Collaborativi di tipo Item-Based. Il suo modello consiste sostanzialmente in una matrice di similarità tra gli item. Nella fase di batch si riceve in input la matrice URM di dimensioni $m \times n$ e si restituisce come output il modello rappresentato da

una matrice $n \times n$ (dove sia sulle righe che sulle colonne sono presenti gli item).

In questo algoritmo gli item sono pensati come vettori nello spazio degli utenti e avranno quindi lunghezza m, dove m rappresenta il numero degli utenti nella matrice URM[12]. La similarità tra due item è misurata calcolando il coseno dell'angolo tra questi due vettori. Formalmente [25] :

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|^2 \times \|\vec{j}\|^2} \tag{3.1}$$

Ogni singola cella $m_{i,j}$ del modello M rappresenterà quindi, il coefficiente di similarità tra l'item i e l'item j. Il coefficiente di similarità tra un item e se stesso sarà pari a 1, mentre se non è presente neanche un utente che ha votato sia l'item i che l'item j, il loro coefficiente di similarità sarà 0.

Nella fase di real-time, la raccomandazione per l'utente u consiste in una lista di N elementi (Top N list), la quale verrà costruita tramite il seguente prodotto vettore-matrice:

$$\text{Raccomandazione } = \text{profilo utente} \times \text{M} \tag{3.2}$$

In questo modo per ogni item i non presente nel profilo utente verrà assegnato un punteggio in base al suo coefficiente di similarità con ogni item $j_{1..n}$ che invece vi è presente. Il punteggio totale per ogni item i sarà la somma tra tutti i coefficienti di similarità con gli item $j_{1..n}$.Questo punteggio, in dataset espliciti, può essere visto come il voto predetto che l'utente darebbe a quell'item. Successivamente si ordina in modo decrescente la lista di item non presenti nel profilo utente in base al punteggio totale ottenuto. La raccomandazione per l'utente u consiste nei primi N item di questa classifica che andranno ha formare la Top-N recommendation list. Il parametro KNN(K-Nearest-Neighbor) indica il numero di item da tenere in considerazione quando, durante la creazione del modello, vengono trovati i coefficentei di similarità trai diversi item. In questo modo si vanno a mantenere i coefficienti tra l'item i e i K item j a lui più vicini (simili), mentre si va ad eliminare tutti quei contributi minimi di similarità tra l'item i e i restanti $n - K$ item meno vicini ad i. Quest'eliminazione parte dal presupposto che tutti i contributi dei film meno simili ad i sono soltanto rumore, per cui una volta eliminati la raccomandazione sarà più affidabile perchè formata considerando solo gli item più simili a quelli presenti nel profilo utente. Il parametro K può variare a seconda della grandezza del dataset, più è ridotto più viene eliminato il rumore, se troppo ridotto però potrebbe portare ad una perdita di contributi che, anche se minimi, potrebbero essere, tutti

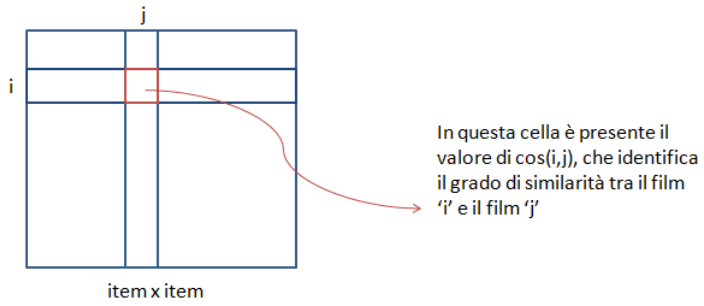


Figura 3.1: Matrice di similarità dell'Item-Item Cosine

insieme tra loro, un coefficiente da tenere in considerazione durante la raccomandazione. Se K è infinito si definisce l'algoritmo soltanto come Item-Item Cosine.

3.4 Algoritmo Item-Item DR

L'Item-Item DR (Direct Relation) è un algoritmo collaborativo di tipo Item-Based. Il modello, come negli algoritmi della classe Item-Item Cosine, consiste in una matrice $n \times n$, di similarità tra gli item. In questo tipo di algoritmo però, la similarità tra un item i e un item j viene calcolata come il numero di occorrenze in cui entrambi gli item i e j sono presenti tra tutti i profili utenti della matrice URM. La costruzione della raccomandazione per questo algoritmo avviene esattamente come per i due precedenti.

3.5 Algoritmo Sarwar

Il Sarwar è un algoritmo collaborativo di tipo Item-Based che utilizza la decomposizione SVD sulla matrice URM, in modo da ridurre lo spazio fisico occupato dal modello perchè viene utilizzata solo una delle tre matrici risultati dalla decomposizioni. Anche in questo algoritmo il modello che viene costruito rappresenta i coefficienti di similarità trai diversi item e viene calcolato nella seguente maniera:

- Si riceva come input la matrice URM di dimensioni $m \times n$ e un parametro ls che indica la dimensione latente dello spazio in cui costruire il modello.

- Si applica la decomposizione SVD alla matrice URM su uno spazio latente di dimensioni ls ottentendo le tre matrici: $U(m \times ls), S(ls \times$

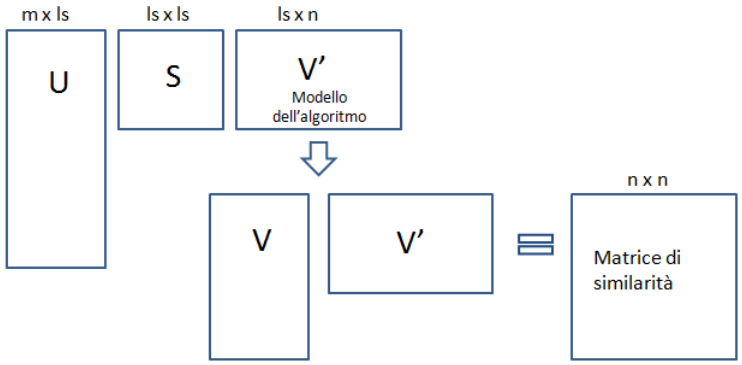


Figura 3.2: Costruzione del modello per l'algoritmo Sarwar e matrice di similarità tra gli item

$ls), V'(ls \times n)$. La matrice V' rappresenta il modello del mio algoritmo ed è, come si può notare, di dimensioni più ridotte rispetto ai modelli costruiti dagli algoritmi Item-Item Cosine dove il modello aveva dimensioni $n \times n$.

- Per risalire alla matrice di similarità tra gli item basta moltiplicare $V \times V'$ che mi da una matrice di dimensioni $n \times n$ con gli item sia sulle righe che sulle colonne.

Per costruire la singola raccomandazione per un utente u, moltiplico il vettore del profilo utente per la matrice di similarità in modo da sommare i coefficienti di similarità riferiti ad ogni singolo item. Quegli item che avranno un coefficiente totale più alto costituiranno la raccomandazione.

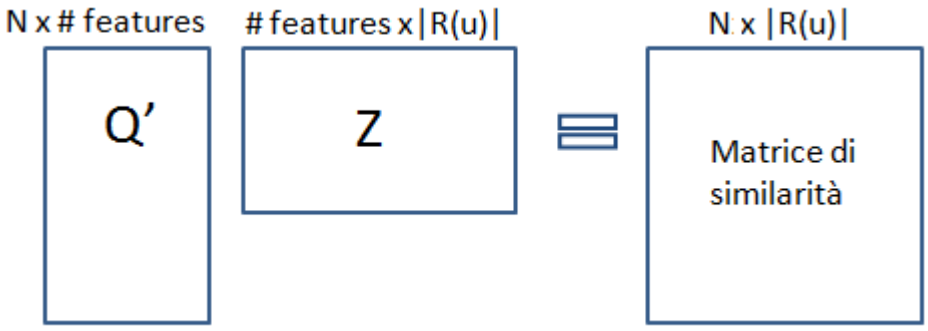


Figura 3.3: Matrice di similarità dell'algoritmo Asymmetric SVD

3.6 Algoritmo Asymmetric SVD

L'Asymmetric SVD è un algoritmo collaborativo di tipo Item-Based sviluppato da Yehuda Koren e vincitore della competizione Netflix Prize. Questo algoritmo lavora sia con dataset impliciti che espliciti. Per ogni item i non presente nel profilo dell'utente u viene calcolato il rating predetto per mezzo della seguente formula[19] [18] :

$$r_{ui} = b_{ui} + q_i^T(\mid R(u) \mid^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj})x_j + (\mid N(u) \mid)^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \quad (3.3)$$

La prima parte della formula è una stima di base con $b_{ui} = \mu + b_u + b_i$ dove μ è la media globale dei rating del dataset, b_u e b_i sono la distanze(bias) dalla media totale μ del dataset rispettivamente dei ratings dati dall'utente u e della media dei ratings ricevuti dall'item i. $\mid R(u) \mid$ è il cardinalità dell'insieme dei ratings espliciti dell'utente u mentre $\mid N(u) \mid$ di quelli impliciti. q, x, y sono tre vettori rispettivamente riferiti agli item i non votati dell'utente, agli item j che hanno ricevuto dall'utente u ratings espliciti e a quelli che hanno ricevuto ratings espliciti. Dalla figura si può vedere come viene ottenuta la matrice di similarità dell'algoritmo. Q' è una matrice di $N \times \#features$, con N lunghezza della raccomandazione e $\#features$ dimensione dello spazio latente, Z ha dimensioni $\#features \times \mid R(u) \mid$ ed è la concatenazione di tutti i $(r_{uj} - b_{uj})x_i + y_i$, questo perchè l'algoritmo è stato applicato su un dataset completamente esplicito e $R(u)$ coincide con $N(u)$. La matrice di similarità è legata alla variabile $\mid R(u) \mid$, dipendente da ogni singolo utente u, per questo la matrice viene creata per ogni singolo utente.

3.6.1 Modifiche apportate all'algoritmo Asymmetric SVD

Durante l'utilizzo sono state apportate le seguenti modifiche all'algoritmo Asymmetric SVD:

- Alcuni parametri utilizzati durante la creazione del modello sono stati cambiati, questo perchè quelli suggeriti dal creatore dell'algoritmo, con un dataset diverso da quello utilizzato durante il Netflix Prize fanno overfitting sul bias degli utenti e dei film, portando l'algoritmo a raccomandare sempre gli stessi film, ovvero quelli con il bias maggiore.
- L'algoritmo Asymmetric SVD durante la raccomandazione real-time fa un ulteriore apprendimento sul profilo utente,il quale è contenuto nel dataset da cui viene creato il modello. Per questo durante i profili ad hoc, avendo utilizzato dei profili esterni alla URM, la fase di apprendimento è stata leggermente modificata in modo da costruire raccomandazione anche ad utenti esterni.

3.7 Algoritmo Content Based: LSA Cosine

LSA Cosine (Latent Semantic Analysis), a differenza degli algoritmi precedenti, è di tipo Content-Based. Il metodo dell'LSA consiste nell'estrarre e rappresentare tramite calcoli statistici l'informazione presente in contenuti testuali [20]. Il concetto generale su cui quest'algoritmo si basa è: l'aggregazione di tutte le parole che compaiono o non compaiono all'interno di un testo stabiliscono un insieme di vincoli di reciprocità che determinano la similarità di significato tra le parole stesse e permettono di raggruppare tra di loro parole simili. L'LSA utilizza, come l'algoritmo Sarwar, la decomposizione SVD, questa volta non ricevendo come input la matrice URM, ma bensì l'ICM, ovvero la matrice contenente tutte le parole che descrivono i diversi item presenti nel dataset che nel caso di film possono essere:

- Parole contenute nel titolo
- Parole contenute nella trama
- Genere
- Cast

3.7.1 Fase Batch: generazione del modello

Essendo un algoritmo Content-Based, l'LSA Cosine costruisce il suo modello in base al contenuto delle risorse. Questi dati (definiti in precedenza come

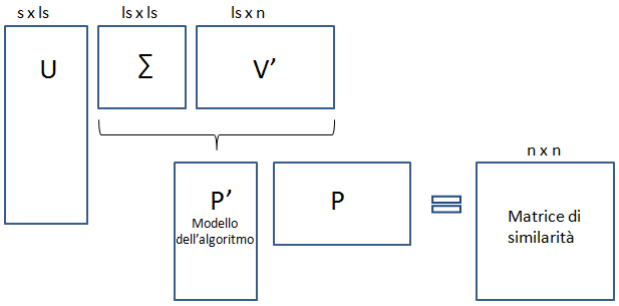


Figura 3.4: Costruzione del modello per l'algoritmo LSA e matrice di similarità tra gli item

stem) sono rappresentati in maniera strutturata nella matrice ICM. Per calcolare quanto uno stem sia importante per un item bisogna valutare due aspetti:

- la frequenza con cui lo stem occorre in un item, più questa è alta più il coefficiente item-stem è alto
- la frequenza con cui lo stem occorre in tutti gli item, più questa è alta più il coefficiente item-stem è basso.

In questo modo, tramite la matrice ICM, rappresentiamo ogni item come vettore nello spazio degli stem, ma dato il numero elevato di questi ultimi, viene prima applicata una decomposizione SVD alla matrice ICM che la taglia utilizzando una specifica dimensione latente k. La decomposizione spezza la matrice ICM in tre matrici U_k, S_k e V_k. La nuova ICM_k, data dal prodotto $S_k \times V'_k$, è pulita dal rumore,integra già le correlazioni trai diversi stem e verrà usata come modello dell'algoritmo. La definizione del parametro k viene solitamente effettuata empiricamente.

3.7.2 Fase Real-Time: generazione della raccomandazione

In questa fase si ricevono come input la matrice URM e il modello generato nella fase precedente. Per costruire la raccomandazione per un singolo utente u, ogni item presente nella matrice URM viene rappresentato con il relativo vettore nello spazio degli stem contenuti nel modello. A questo punto, così come avveniva per gli algoritmi Collaborativi Cosine e Cosine KNN, viene calcolata la similarità tra gli item i_m presenti in un profilo utente e gli item j_n che invece non sono presenti, tramite il coseno dell'angolo tra il vettore $\vec{u}$ che rappresenta l'utente e quelli degli item j_n. Più l'angolo è piccolo più l'item i rappresentato dal vettore $\vec{j}$ sarà simile ai gusti dell'utente, per questo

vengono raccomandati gli$N$ (Top-N list) item j_n i cui vettori hanno l'angolo più piccolo dal vettore $\vec{u}$.

Capitolo 4

Descrizione del problema e verifica esistenza

All'interno di questo capitolo descriveremo il problema della presenza di Reshuffling (o Diversity), i dataset utilizzati, la metodologia di verifica dell'esistenza del problema e i risultati ottenuti da questa fase di verifica.

4.1 Descrizione del problema

Prima di descrivere il vero e proprio problema è necessario introdurre diversi concetti. In un sistema di raccomandazione i dati su cui si basano le vere e proprie raccomandazioni sono in continua evoluzione. Facendo riferimento all'ambito dell'IPTV, e più precisamente al Video on Demand (VoD), le continue richieste di film da parte degli utenti ed il continuo inserimento di rating (espliciti o impliciti) all'interno del sistema portano ad influenzare le raccomandazioni stesse più o meno pesantemente. Tali raccomandazioni sono infatti basate su un modello che, a seconda della frequenza di aggiunta di rating, di nuovi utenti e nuovi item, deve essere continuamente aggiornato (e quindi ricostruito) in modo da mantenere una determinata accuratezza del servizio. Se si pensa ad un singolo utente, supponendo che in due istanti di tempo successivi t_1 e t_2 (con $t_1 < t_2$) venga richiesta una raccomandazione, si potrà spesso notare, proprio per i motivi sopracitati, che le due liste di film raccomandati saranno diverse tra loro. Questa differenza, valutata in termini di numero di film cambiati tra la prima e la seconda lista, prende il nome di Reshuffling (o Diversity). Date due liste di raccomandazione (top N), L_1 e L_2, formalmente la differenza tra di esse (e quindi il valore di Reshuffling o di diversità) è definita come i film presenti in L_2 che non

compaiono in L_1 [21]:

$$L_2 \setminus L_1 = \{x \in L_2 | x \notin L_1\} \tag{4.1}$$

Questa differenza può anche essere espressa in termini percentuali semplicemente dividendola per N (dove N consiste nella lunghezza delle liste di raccomandazione) [21]:

$$\text{Reshuffling}(L_1; L_2; N) = \frac{|L_2 \setminus L_1|}{N} \tag{4.2}$$

Se L_2 e L_1 sono identiche, il Reshuffling (o Diversity) sarà pari a 0. Se L_2 e L_1 sono completamente differenti, il Reshuffling sarà pari a 1. Il Reshuffling è causato da delle perturbazioni all'interno del sistema di raccomandazione. Con il termine *perturbazione* intendiamo sostanzialmente una modifica dei dati interni al sistema per poter influenzare le raccomandazioni fornite agli utenti. Vi sono due tipi di perturbazioni: naturali e forzate. Le perturbazioni naturali sono principalmente dovute all'evoluzione temporale dei dati:

- Evoluzione temporale del profilo utente dovuta all'arricchimento del proprio profilo utente con nuove votazioni.
- Evoluzione del modello su cui si basa la raccomandazione in seguito all'aggiornamento del modello stesso da parte del sistema (potrebbero essere stati inseriti nuovi voti, nuovi utenti e nuovi item).

Le perturbazioni forzate, invece, non derivano dall'evoluzione temporale dei dati ma consistono spesso in interventi guidati e soprattutto voluti, come ad esempio attacchi al sistema o algoritmi per la spiegazione delle raccomandazioni etc.in ogni caso non saranno argomento di questo lavoro di tesi. In un primo momento il Reshuffling può apparire come una conseguenza naturale e positiva dell'azione real-time dei sistemi di raccomandazione, ma non è sempre così. Se da un lato, infatti, il presentare sempre la stessa lista di raccomandazione (raccomandazione statica) non è assolutamente ammissibile proprio perchè di questa azione a real-time non ve ne sarebbe traccia e ciò renderebbe il sistema di raccomandazione pressochè inutile agli occhi dell'utente, da un altro un'eccessiva presenza di Reshuffling (quindi un'eccessiva dinamicità o diversità delle liste di raccomandazione) potrebbe anch'essa sfiduciare l'utente nei confronti del sistema, poichè, ad esempio, il sistema potrebbe dare l'impressione di consigliare i film in maniera casuale. Questo lavoro di tesi si concentra proprio su quest'ultimo caso, con l'obiettivo di andare a limitare il fenomeno di Reshuffling, cercando di dare,

quindi, una certa regolarità alle raccomandazioni, ovviamente evitando di renderle troppo statiche. L'eccessiva presenza di Reshuffling può essere data da diversi fattori:

- Quando l'utente che richiede una raccomandazione ha un profilo composto da pochi rating: l'aggiunta di un rating, infatti, si sintetizza in una modifica drastica delle sue preferenze.
- Quando il tempo tra due sessioni utente aumenta (il tempo che intercorre tra due richieste di raccomandazione). Questo è sostanzialmente dovuto alla modifica del modello su cui si basano le raccomandazioni stesse.
- Durante il primo avvio del sistema (cold-start stage).

Per gli algoritmi di tipo collaborativo, poi, sorge un altro fattore di notevole importanza che consiste nella presenza di film popolari all'interno del profilo di un utente (o comunque all'aggiunta di film popolari al profilo utente durante una sessione). Per film popolari si intendono tutti quei film che hanno un elevato numero di voti e che possono maggiormente influire sulla raccomandazione. Un film popolare, infatti, ha la caratteristica principale di essere simile ad un numero elevato di film, proprio perchè è stato votato da molti utenti diversi. Di conseguenza può influire sul calcolo della matrice di similarità e quindi può influire sulla raccomandazione stessa. Nel nostro caso, in seguito ad analisi effettuate sui dataset, è stata ricavata la lista dei film popolari nel seguente modo: sono stati ordinati i film in base al numero totale di rating ricevuti da parte degli utenti. Partendo dal film con più voti, l'insieme di film con un numero di rating totale (somma del numero di rating ricevuto da ogni film) superiore al 50% dei rating totali presenti nel dataset corrisponde all'insieme dei film popolari. Ad esempio Supponiamo che il sistema fornisca unicamente 4 film f_1, f_2, f_3, f_4 con rispettivamente 40, 30, 20, 25 voti. Il primo passo è di ordinarli secondo il numero di voti, otterremo quindi il seguente ordine: f_1, f_2, f_4, f_3. Il numero totale dei voti è pari a 115, di conseguenza il 50% è pari a 57.5. I primi due film (f_1 e f_2) rappresentano l'insieme dei film popolari in quanto la somma dei loro voti è 70 che è maggiore di 57.5.

E' chiaro che, durante l'evoluzione di un sistema di raccomandazione, l'insieme dei film popolari può cambiare, sia in termini di cardinalità sia in termini di elementi che lo compongono. Durante la nostra analisi però, avendo a disposizione dataset statici o comunque dataset temporali di breve periodo, si sono sempre mantenuti gli stessi film popolari.

4.2 Dataset

Per attenersi a casi reali si sono utilizzati tre dataset di diverse dimensioni: due dataset italiani forniti da servizio di Video on Demand anonimo che chiameremo d'ora in avanti MovieRec e un dataset americano relativo al servizio di videonoleggio fornito da Netflix, questo dataset è stato reso pubblico grazie alla competizione indetta da Netflix per lo sviluppo di nuovi sistemi di raccomandazione . Più precisamente, i due dataset MovieRec, sono rispettivamente uno statico e uno temporale. Nel primo non vi sono informazioni circa gli istanti temporali in cui sono stati inseriti i rating da parte degli utenti, ma soltanto. Temporale nel senso che all'interno del dataset, per ogni voto, sono presenti informazioni riguardo all'istante temporale in cui questo rating è stato inserito. In questo modo è possibile ripercorrere la reale costruzione di ogni profilo del dataset.Il dataset di Netflix è anch'esso statico. I dataset statici verranno utilizzati per testare la risoluzione del problema di Reshuffling tramite metologie di analisi statistiche e personalizzate utilizzando i rating senza nessun ordinamento temporale. Il dataset temporale verrà utilizzato nella fase finale di test per valutare la qualità della tecnica di risoluzione in un determinato periodo di tempo (simulando, quindi, un vero e proprio sistema di raccomandazione). Qui di seguito verranno presentate le caratteristiche tecniche di ogni singolo dataset.

	N° Utenti	N° Film	N° Rating	Densità	N° film popolari
Dataset MovieRec	50889	985	321783	0,64%	97
Dataset Netflix	126603	6890	1373697	0,15%	290
Dataset temporale MovieRec	70771	773	558803	1,02%	127

Figura 4.1: Caratteristiche dataset utilizzati

4.2.1 Dataset statico MovieRec

Questo dataset è formato da rating impliciti, i quali, di conseguenza, hanno valore 0 oppure 1. Non avendo informazioni temporali non si può evincere la quantità giornaliera di rating. Le dimensioni della URM sono pari a 50889 utenti per 985 film. Il numero di rating presenti all'interno di questa matrice è pari a 321783. Proprio per il fatto che i profili poco lunghi sono quelli che possono essere più affetti dal problema di Reshuffling è interessante analizzare la distribuzione degli utenti in base alla lunghezza del profilo. Questa analisi è stata effettuata su tutti e tre i dataset e ha portato alla

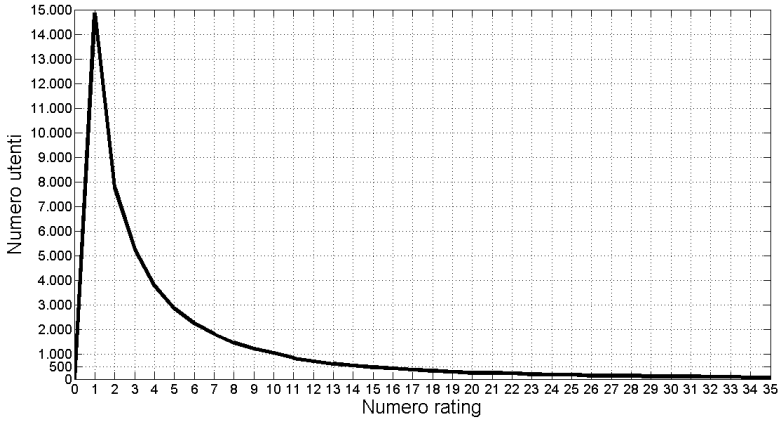


Figura 4.2: Dataset MovieRec: distribuzione utenti per lunghezza del profilo

definizione di categorie di utenti (suddivise in base alla lunghezza del profilo) che verranno trattate in seguito.

Come si può notare dalla Figura 4.2, la stragrande maggioranza degli utenti ha un profilo che contiene al massimo dieci rating. Tra questi poi, la maggior parte ha al massimo quattro rating, ossia vi sono profili molto corti all'interno di questo dataset. Se andiamo ad osservare la curva oltre dieci rating notiamo che vi sono si molti profili (si parla comunque di migliaia di profili utente) ma che pian piano questa curva tende asintoticamente a zero. In ogni caso questa parte di utenti è comunque molto minore rispetto agli utenti con profili con meno di dieci rating. Dalla matrice URM a disposizione è stata effettuata anche un'analisi della popolarità dei film.

Come si nota dalla Figura 4.3 i film popolari, invece, corrispondono all'incirca al 9.85% dei film totali, pari a 97 film su 985. La somma dei rating di questi film è superiore al 50% dei rating totali presenti nell'intero dataset.

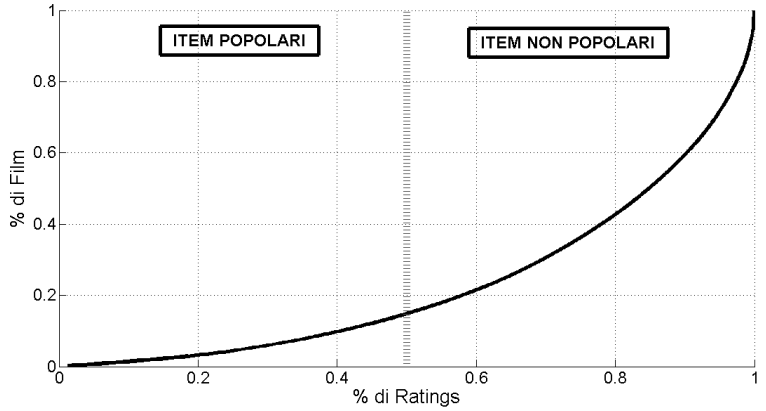


Figura 4.3: Dataset MovieRec: percentuale di voti in base alla percentuale di film

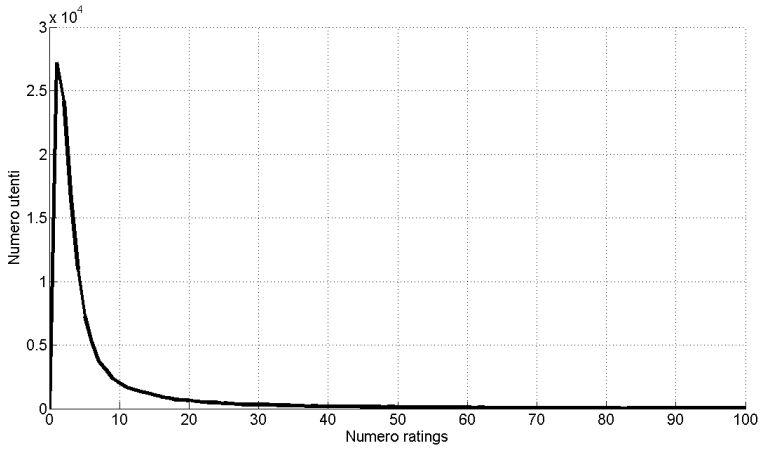


Figura 4.4: Dataset Netflix: distribuzione utenti per lunghezza del profilo

4.2.2 Dataset statico Netflix

Questo dataset contiene rating di tipo esplicito. La scala di rating varia da 1 a 5. Anche qui non si hanno informazioni di tipo temporale, dunque non si può evincere la frequenza di rating giornaliera. Le dimensioni della URM sono pari a 126603 utenti per 6890 films. Il numero di rating presenti all'interno di questa matrice è pari a 1373697. La distribuzione di utenti per lunghezza del profilo evidenzia il fatto che anche qui la stragrande maggioranza degli utenti ha dei profili corti, inferiori o pari a dieci rating (vedi Figura 4.5.

Come si nota dalla Figura 4.6 i film popolari corrispondono al 4,2 % dei film totali del dataset, più precisamente 290 film su 6890.

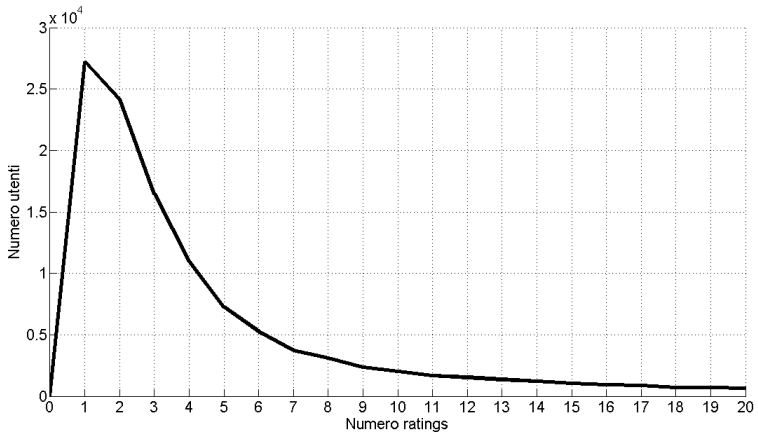


Figura 4.5: Dataset Netflix: distribuzione utenti per lunghezza del profilo al massimo di 20 rating

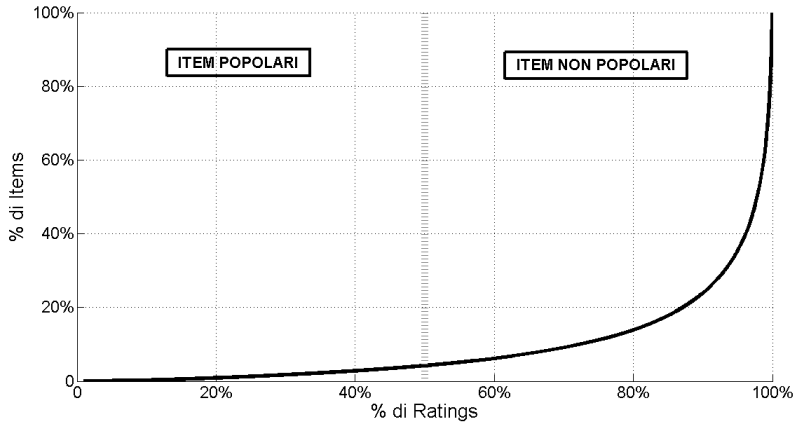


Figura 4.6: Dataset Netflix: percentuale di voti in base alla percentuale di film

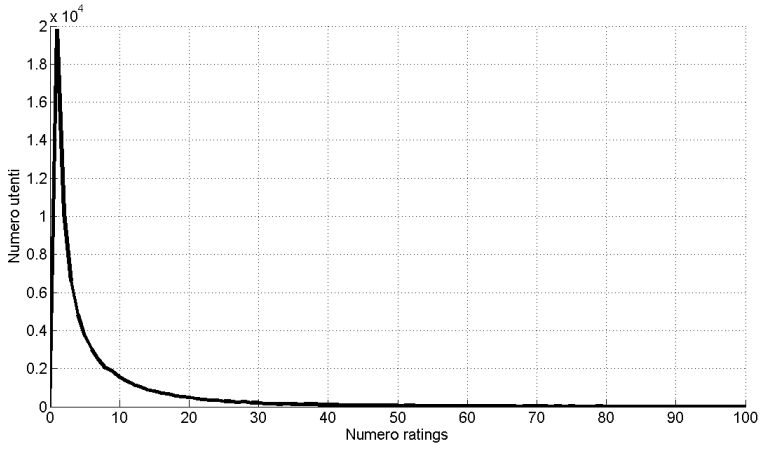


Figura 4.7: Dataset MovieRec temporale: distribuzione utenti per lunghezza del profilo

4.2.3 Dataset temporale MovieRec

Questo dataset, contenente rating impliciti, a differenza dei due precedenti ha memorizzate le informazioni sugli istanti di tempo in cui i rating sono stati aggiunti al sistema. Per questo dataset, quindi, oltre alle informazioni sugli utenti e sui film popolari è possibile ottenere anche dei dati sulla frequenza con cui i rating vengono aggiunti al sistema.
Le dimensioni della URM sono pari a 70771 utenti per 773 film. Il numero di rating presenti all'interno di questa matrice è pari a 558803. La distribuzione di utenti per lunghezza del profilo evidenzia (come si vede in Figura 4.7) il fatto che la stragrande maggioranza degli utenti ha dei profili lunghi, al massimo, 20 rating. Tra questi, la maggioranza ha profili lunghi al massimo 10 rating.

L'analisi relativa agli item popolari, come si vede in Figura 4.8, porta a concludere che i film popolari sono pari a 127, all'incirca il 16.4% dei film presenti all'interno del dataset.

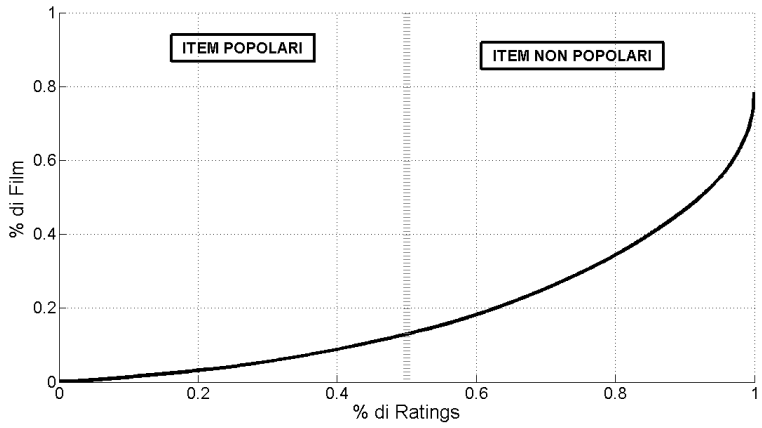


Figura 4.8: Dataset MovieRec temporale: percentuale di voti in base alla percentuale di film

Come si può notare dalla Figura 4.9, la frequenza settimanale di aggiunta dei rating, in media, si aggira intorno a 1.7 rating a settimana per utente. Dalla Figura 4.10, invece, si nota che la frequenza mensile si aggira , mediamente, attorno a 2.5 rating a settimana per utente. La bassa differenza tra i due valori significa che probabilmente ci sono alcuni utenti che inseriscono molti rating assieme all'interno di una settimana e non ne inseriscono più per il resto del mese, un comportamente del genere fa si che la frequenza mensile non sia, come si potrebbe immaginare, 4 volte quella settimanale. Per misurare la frequenza di aggiunta dei rating sono stati considerati soltanto gli utenti attivi nei diversi intervalli di tempo. Un utente viene considerato attivo se esprime almeno un rating in un determinato intervallo di tempo.

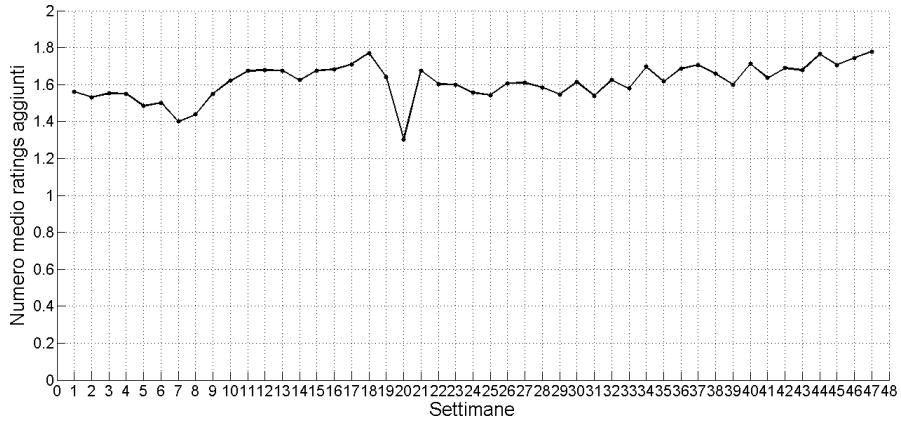


Figura 4.9: Frequeza settimanale di aggiunta dei rating nel dataset

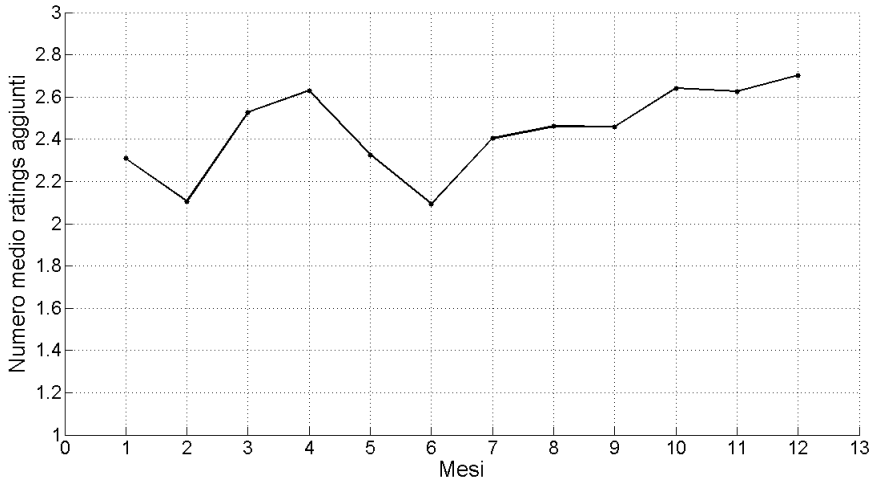


Figura 4.10: Frequeza mensile di aggiunta dei rating nel dataset

4.3 Verifica dell'esistenza del problema

Verranno presentate qui di seguito le metodologie e i risultati ottenuti durante la fase di verifica dell'esistenza del problema.

4.3.1 Prima metodologia di verifica

Il primo passo è stato quello di individuare delle particolari categorie di utenti all'interno dei due dataset statici utilizzati in questa fase. Gli utenti sono stati suddivisi in base alla lunghezza del profilo (numero di rating presenti in ogni profilo utente). Questo procedimento, infatti, era necessario per riuscire a capire l'esistenza di tale problema a seconda della lunghezza del profilo utente. Ci si aspetta che per profili utente di lunghezza ridotta si presenti un'elevata dinamicità, al contrario per i profili composti da un gran numero di rating ci sia una bassa o addirittura assente dinamicità. Sono state individuate 5 categorie:

1. Utenti con profili di lunghezza tra 1 e 5 rating (compresi gli estremi)
2. Utenti con profili di lunghezza tra 6 e 10 rating (compresi gli estremi)
3. Utenti con profili di lunghezza tra 11 e 20 rating (compresi gli estremi)
4. Utenti con profili di lunghezza tra 21 e 40 rating (compresi gli estremi)
5. Utenti con profili di lunghezza oltre i 40 rating

Con un numero inferiore di rating abbiamo utilizzato un intervallo minore, data la maggior percentuale di utenti appartenenti a queste fasce,mentre mano a mano che i rating aumentano abbiamo ingrandito gli intervalli dato che le percentuali di utenti appartenenti a queste fasce si riducono gradualmente. La fascia degli utenti con nessun voto è stata subito scartata perché non utile ai fini di questo test.

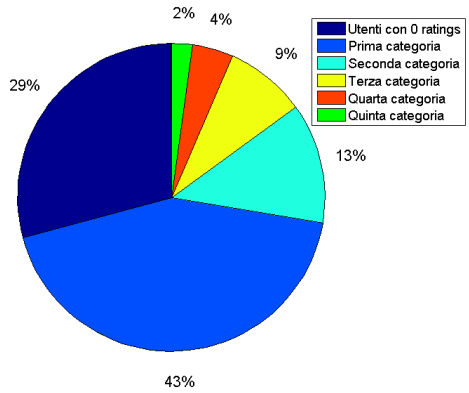


Figura 4.11: Dataset MovieRec: percentuali utenti per categoria

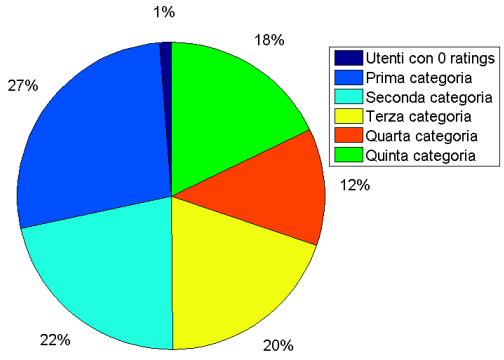


Figura 4.12: Dataset Netflix: percentuali utenti per categoria

Il test è stato svolto su tutti gli utenti del dataset per ogni categoria precedentemente presentata verificando la presenza di Reshuffling aggiungendo un voto ad ogni singolo profilo e verificando il valore di Reshuffling. Siccome non si può essere sicuri riguardo l'affidabilità del valore di Reshuffling, si è deciso di assumere che tali valori seguissero una distribuzione normale con media pari alla media dei valori di Reshuffling e varianza nota. In base a questa assunzione si è deciso di costruire un intervallo di confidenza per la media attraverso la varianza stessa, più precisamente tramite l'errore standard. Utilizzando la seguente formula :

$$\text{intervallo di confidenza} = \text{media} \pm (2 \times \text{errore standard}) \tag{4.3}$$

abbiamo costruito un intervallo di confidenza per la media al novantacinquesimo percentile. La media è calcolata come:

$$\text{media} = \frac{\sum_{i=1}^{n} \text{Valore di Reshuffling}_i}{n} \tag{4.4}$$

L'errore standard è calcolato come:

$$\text{errore standard} = \frac{\text{deviazione standard}}{\sqrt{n}} \tag{4.5}$$

La deviazione standard è calcolata come :

$$\text{deviazione standard} = \sqrt{\frac{\sum_{i=1}^{n} (\text{Valore di Reshuffling}_i - \text{media})^2}{n-1}} \tag{4.6}$$

Dato che la verifica della presenza di Reshuffling prevede l'aggiunta di un voto per ogni profilo testato, è stato effettuato una sorta di test inverso. Anziché partire da un profilo del dataset e aggiungere casualmente un rating, si è deciso di togliere un rating già esistente all'interno del profilo stesso, in modo da utilizzare tale nuovo profilo come profilo di partenza. In questo modo il rating che viene aggiunto al profilo non viene scelto a caso ma, essendo presente nel dataset originale, si attiene ad un caso reale. In pratica la presenza di Reshuffling viene valutata, per ogni profilo, seguendo l'evoluzione reale del profilo stesso. I ogni caso avendo utilizzato dataset statici, il rating che viene tolto non è l'ultimo in sequenza temporale ma viene soltanto preso a caso. Questo procedimento inverso di testing ha implicato che la composizione delle categorie cambiasse: ogni profilo è stato riscalato come se avesse un rating in meno rispetto al caso reale. Tutti quei profili che nel dataset originale avevano un rating, nel nuovo dataset verranno considerati come appartenenti alla categoria nessun voto, e così via. Durante l'esecuzione di questi test, è stato considerato anche l'effetto

degli item popolari (anche nell'algoritmo di tipo Content-Based) sulle varie raccomandazioni. Inoltre i risultati sono stati suddivisi andando anche a considerare la popolarità di un profilo utente, definita come segue: un profilo utente è popolare se contiene almeno un film popolare. Si rimanda la lettura della sezione successiva per i risultati del sopracitato test.

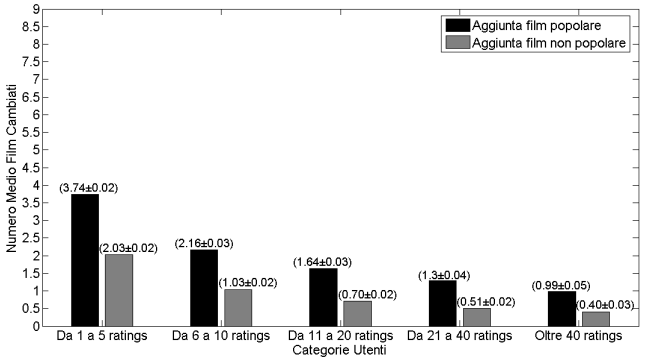
4.3.2 Risultati prima metodologia di verifica

Sono stati riportati nelle Figure 4.13(a), 4.13(b), 4.14(a), 4.14(b) i risultati ottenuti per le cinque categorie trattate utilizzando l'algoritmo Sarwar sul dataset MovieRec statico e l'algoritmo Asymmetric SVD sul dataset Netflix. Dalle figure possiamo evidenziare come la presenza di Reshuffling, come da aspettative, sia influenzata dalla lunghezza del profilo utente. Più il profilo è lungo più il valore di Reshuffling tende a diminuire. Le categorie più soggette a questo problema sono, sicuramente, le prime due. Per questo si è deciso di concentrare l'attenzione sui profili utente che rientrassero in queste due categorie. Nelle Figure 4.15(a), 4.15(b), 4.16(a), 4.16(b), 4.17(a), 4.17(b), 4.18(a), 4.18(b) sono riportati i risultati ottenuti in merito a queste categorie per ogni algoritmo di raccomandazione utilizzato. Dalle figure possiamo ricavare le seguenti considerazioni:

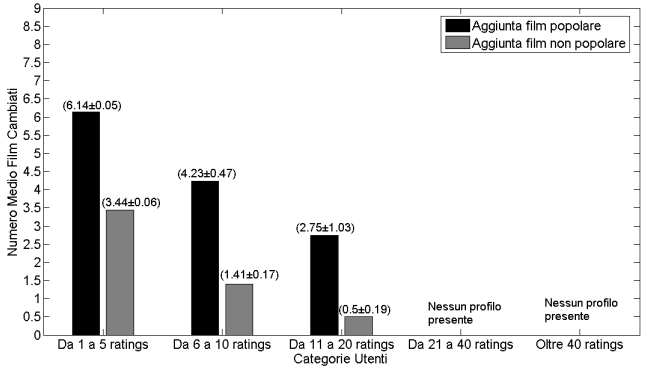
- Il primo risultato che si può facilmente evidenziare è che la prima categoria, per tutti gli algoritmi considerati, è la più affetta dal problema di Reshuffling.

- La popolarità del profilo incide significativamente sul valore di Reshuffling, l'analisi su profili popolari (che contengono uno o più film popolari) evidenzia l'influenza che essi hanno sulla raccomandazione, la loro presenza infatti implica una minore dinamicità della lista. I profili non popolari, al contrario, risultano maggiormente affetti da questo problema. Di conseguenza possiamo dedurre che una raccomandazione per un profilo popolare resterà più fedele al profilo utente stesso. Il Reshuffling per un profilo non popolare arriva a volte ad essere il doppio rispetto a quello ottenuto con un profilo popolare.

- La popolarità del film aggiunto incide sul valore di Reshuffling, valori più alti si ottengono quando viene aggiunto al profilo un film popolare. Ovviamente la popolarità del film aggiunto va a scontrarsi con la popolarità del profilo utente: nel caso in cui il profilo di partenza è non popolare abbiamo una dinamicità della lista, a volte superiore al 60%, maggiore rispetto ai casi in cui il profilo di partenza è popolare.

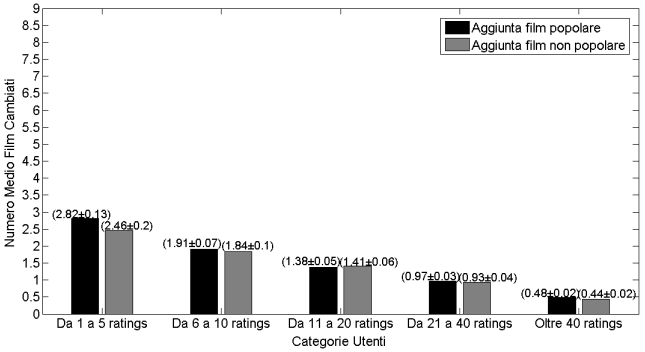


(a) Profilo popolare

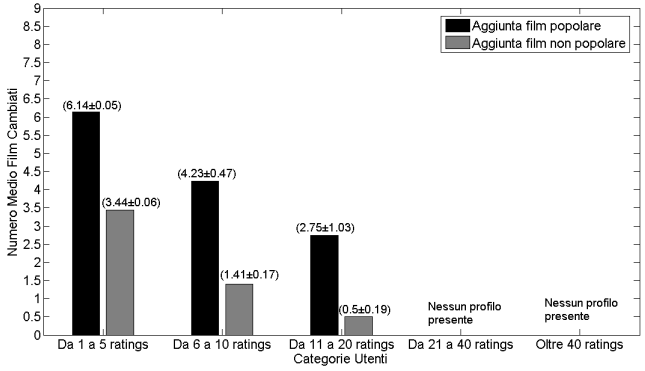


(b) Profilo non popolare

Figura 4.13: Dataset MovieRec - Algoritmo Sarwar: Valore di Reshuffling per profilo di partenza popolare (a) e non popolare (b)

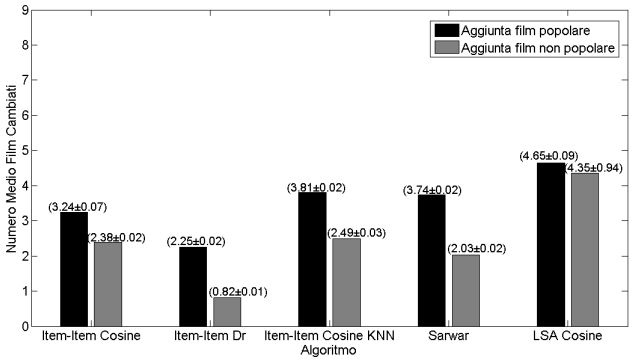


(a) Profilo popolare

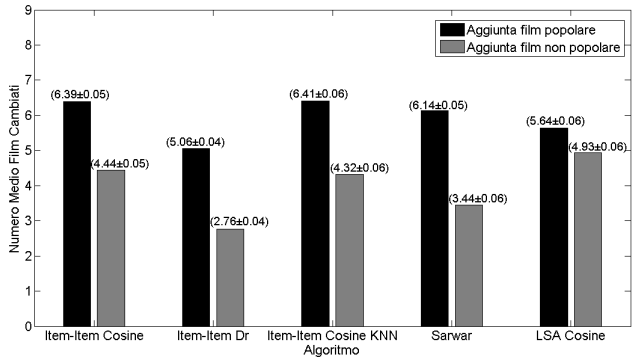


(b) Profilo non popolare

Figura 4.14: Dataset Netflix - Algoritmo Asymmetric SVD: Valore di Reshuffling per profilo di partenza popolare (a) e non popolare (b)

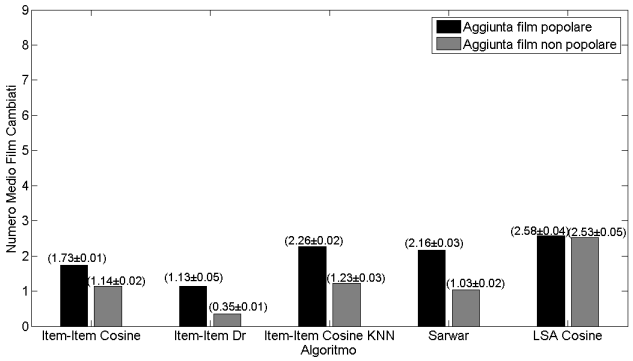


(a) Profilo popolare

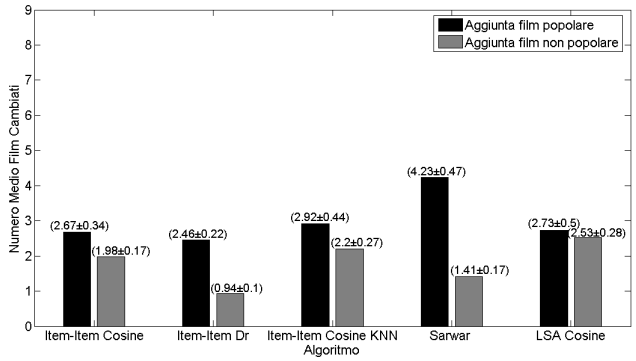


(b) Profilo non popolare

Figura 4.15: Dataset MovieRec: Valore di Reshuffling per la prima categoria per profilo di partenza popolare (a) e non popolare (b)

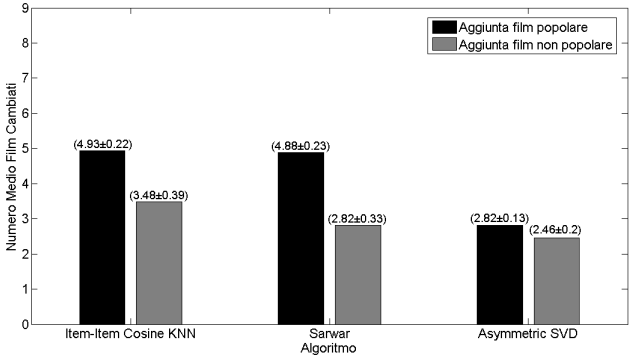


(a) Profilo popolare

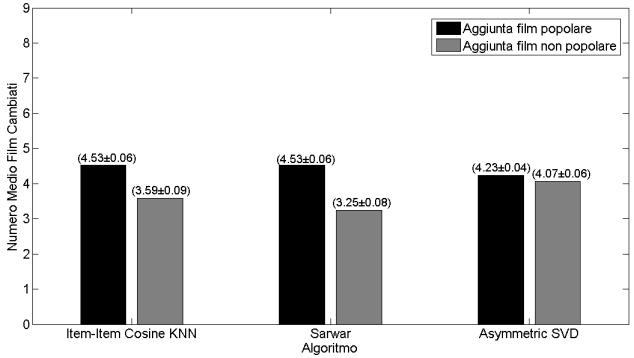


(b) Profilo non popolare

Figura 4.16: Dataset MovieRec: Valore di Reshuffling per la seconda categoria per profilo di partenza popolare (a) e non popolare (b)

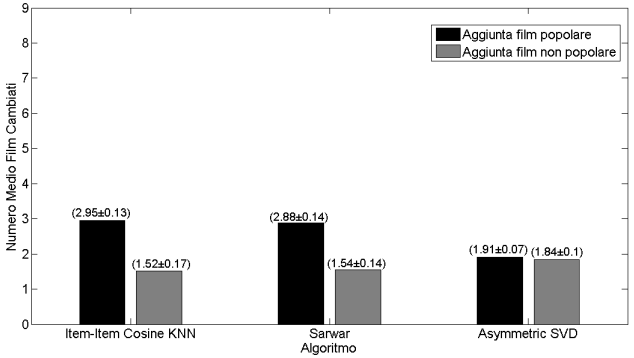


(a) Profilo popolare

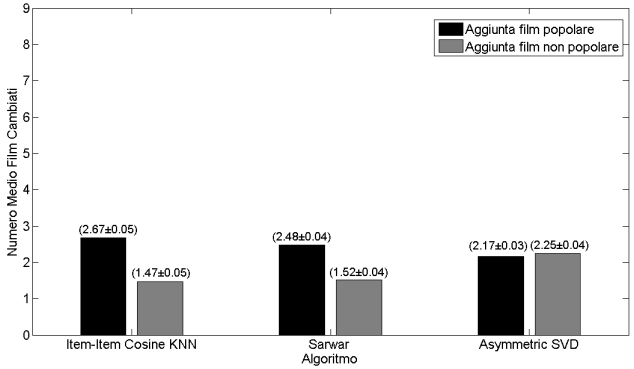


(b) Profilo non popolare

Figura 4.17: Dataset Netflix: Valore di Reshuffling per la prima categoria per profilo di partenza popolare (a) e non popolare (b)



(a) Profilo popolare



(b) Profilo non popolare

Figura 4.18: Dataset Netflix: Valore di Reshuffling per la seconda categoria per profilo di partenza popolare (a) e non popolare (b)

- L'algoritmo che presenta meno Reshuffling è l'Item-Item DR, questo potrebbe derivare dal fatto che questo algoritmo da molta più attenzione ai film che hanno più relazioni con altri film all'interno del modello. Questo risultato si ripercuote sulla raccomandazione portando a consigliare frequentemente questa elite di film, che raramente cambia al variare del profilo utente. Solitamente i film più relazionati sono quelli popolari.
- L'algoritmo LSA, essendo Content-Based, presenta poca differenza tra il valore di Reshuffling a seconda che il profilo o l'item aggiunto sia popolare o non popolare.
- Gli altri algoritmi presentano dei valori di Reshuffling abbastanza simili che variano dal 20% al 60%.

4.3.3 Seconda metodologia di verifica: profili ad hoc

In questa seconda fase di test sono stati costruiti dei profili ad hoc, in modo da verificare la presenza di Reshuffling (e la qualità della raccomandazione) in particolari situazioni (come ad esempio nel caso di un utente appassionato ai film di Spielberg). Se da un lato la prima tipologia di test è di natura empirica e va a verificare unicamente la presenza di Reshuffling in determinate situazioni, dall'altro i profili ad hoc permettono di ottenere una valutazione soggettiva della qualità della raccomandazione. E' logico che, nell'ambito della valutazione del metodo di risoluzione, non basterà una misura soggettiva (bisognerà fare riferimento anche a metriche come la Recall etc.).

Inizialmente sono stati costruiti otto profili composti da diversi film i quali avevano tutti un fattore in comune (genere, regista etc.). Sono state poi stilate una serie di aspettative relative sia al valore di Reshuffling che ci si sarebbe aspettati sia relative alla qualità della raccomandazione (sostanzialmente alle tipologie di film che ci si sarebbe aspettati di avere all'interno della lista di raccomandazione). L'esecuzione del test prevedeva, poi, di aggiungere una serie di film, più o meno inerenti al profilo stesso (ad esempio in un profilo contenente solo cartoni animati, veniva aggiunto in un primo momento un cartone animato e in un secondo momento un horror o un film con caratteristiche completamente diverse) e di valutare i risultati della raccomandazione rifacendosi alle aspettative. Il valore di Reshuffling è misurato come nel test precedente. Si rimanda la lettura al paragrafo successivo per i risultati del sopracitato test.

4.3.4 Risultati seconda metodologia di verifica: profili ad hoc

Verranno qui di seguito presentati i risultati dei test sui due dataset statici sopracitati. Ogni singola azione è effettuata sul profilo iniziale ed è indipendente dalle altre azioni. I Test fanno riferimento a liste di raccomandazione di dieci film. Prima di elencare i risultati è bene introdurre le seguenti definizioni:

- Per Reshuffling minimo **(MIN)** si intende un cambiamento della lista di raccomandazione al più pari al 20% (esempio: in una Top-10 al massimo due item possono cambiare per rimanere nella fascia di Reshuffling minimo).

- Per Reshuffling medio **(MED)** si intende un cambiamento della lista di raccomandazione tra il 30% e il 50%.

- Per Reshuffling alto **(MAX)** si intende un cambiamento della lista di raccomandazione superiore al 50%.

All'interno della tabella verranno anche indicate delle linee guida per valutare la qualità della raccomandazione. Nella colonna aspettative, oltre al valore di Reshuffling che ci si aspetta verrà anche indicato le caratteristiche dei film consigliati, nella maggior parte il genere. Ad esempio con la dicitura (Cartoni animati) ci si aspetterà una raccomandazione con prevalenza di cartoni animati. Nella colonna di ogni algoritmo verrà poi indicato se questa previsione verrà rispettata oppure o no (semplicemente con una dicitura (SI) o (NO)). Per ogni cella relativa ad uno specifico algoritmo verrà assegnato un colore: se la cella è grigia significa che le aspettative (in merito al valore di Reshuffling) non sono state rispettate, se è bianca significa che il valore di Reshuffling che ci aspettavamo è stato rispettato o è addirittura migliore. Il significato delle sigle contenute nelle tabelle è il seguente:

L Lunghezza del profilo, numero di rating

FC Fattore Comune che raggruppa i film contenuti nel profilo

R Reshuffling

Tabella 4.1: Dataset MovieRec: risultati profili ad hoc (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni	Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
1	-**L**: 5 -**FC** : Genere	MIN (Stesso genere)	Stesso genere(POP)	R:30% (SI)	R:10% (SI)	R:10% (NO)	R:10% (SI)	R:40% (SI)
		MIN (Stesso genere)	Altro genere(NOPOP)	R:20% (SI)	R:20% (SI)	R:20% (NO)	R:0% (SI)	R:0% (SI)
2	-**L**: 3 -**Profilo non popolare** -**FC**: Attore , Genere	MIN (Stesso genere)	Stesso genere e attore (NOPOP)	R:10% (SI)	R:10% (SI)	R:10% (SI)	R:10% (SI)	R:20% (SI)
		MIN (Stesso genere)	Stesso genere(NOPOP)	R: 0% (SI)	R:10% (SI)	R:20% (SI)	R:10% (SI)	R:30% (SI)
		MIN-MED (Stesso genere)	Altro genere, altro attore (NOPOP)	R:10% (SI)	R:10% (SI)	R:30% (SI, però presenza di film d'azione)	R:10% (SI)	R:10% (SI)
3	-**L**: 7 -**FC**: Genere	MIN (Stesso genere)	Stesso genere (NOPOP)	R:10% (SI)	R: 20% (SI)	R:0% (SI)	R:10% (SI)	R:10% (SI)
		MIN (Stesso genere)	Altro genere (NOPOP)	R:0% (SI)	R:0% (SI)	R:10% (SI)	R:0% (SI)	R:0% (SI)
4	-**L**: 4 -**FC**: Argomento trattato	MIN (Stesso argomento)	Stesso argomento (NOPOP)	R:20% (NO)	R:50% (NO)	R:10% (NO)	R:20% (NO)	R:20% (NO)
		MIN (Stesso argomento)	Altro argomento (NOPOP)	R:10% (NO)	R:10% (NO)	R:0% (NO)	R:10% (NO)	R:10% (NO)

Tabella 4.2: Dataset MovieRec: risultati profili ad hoc (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni	Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
5	-**L**: 4 -**Profilo non popolare** -**FC**: Regista	MIN	Stesso regista (NOPOP)	R:40%	R:50%	R:40%	R:70%	R:30%
		MIN	Altro regista (POP)	R:40%	R:60%	R:30%	R:10%	R:70%
6	-**L**: 3 -**Profilo non popolare** -**FC**: Generi, sequel	MIN (Stessi generi)	Stesso sequel (NOPOP)	R:30% (SI)	R:30% (SI)	R:20% (NO)	R:60% (NO)	R:30% (NO)
		MIN (Stessi generi)	Stesso generi, no sequel (NOPOP)	R:10% (SI)	R:10% (SI)	R:0% (NO)	R:10% (NO)	R:30% (NO)
		MIN-MED (Stessi generi)	Altro genere, no sequel (NOPOP)	R:0% (SI)	R:0% (SI)	R:30% (NO)	R:30% (NO)	R:0% (NO)
7	-**L**: 5 -**FC**: Popolarità	MIN	(POP)	R:20%	R:20%	R:0%	R:30%	R:10%
		Nessun cambiamento	(NOPOP)	R:0%	R:0%	R:0%	R:0%	R:100%
8	-**L**: 5 -**Profilo non popolare** -**FC**: Non popolarità	Nessun cambiamento	(NOPOP)	R:0%	R:0%	R:0%	R:0%	R:30%
		MED	(POP)	R:80%	R:60%	R:80%	R:80%	R:10%

Test sui profili ad hoc di MovieRec: discussione

L'esito dei precedenti test, mostrato nelle Tabelle 4.1 e 4.2, ha sottolineato come, nella maggior parte dei casi, vengano rispettate le previsioni in merito al grado di presenza di Reshuffling. I casi che sforano dalla previsione di Reshuffling sono comunque molteplici, e spesso superano di molto le aspettative, generando delle variazioni che portano anche a cambiamenti del 90-100% delle intere liste di raccomandazione.I risultati peggiori si ottengono nel profilo con Regista come fattore comune (profilo 5), dove tutti gli algoritmi superano il valore di Reshuffling atteso, nel profilo contenente dei sequel (profilo 6) dove soltanto l'algoritmo Item-Item DR (quello in generale meno influenzato dal Reshuffling) presenta una bassa dinamicità e nel profilo non popolare (profilo 8), nel quale una volta aggiunto un film popolare, la lista cambia per più della metà, questo probabilmente perchè un film popolare, in algoritmi di tipo collaborativo può azzerare il contributo degli altri film presenti nel profilo. Ovviamente questo discorso non vale per l'algoritmo Content-Based, dove la popolarità di un film non influisce in nessun modo con la raccomandazione. Il risultato complessivo di questo test è comunque buono, ci si aspettavano risultati peggiori.

Per quanto riguarda la qualità della raccomandazione, oltre la metà dei casi danno esiti positivi (39 casi su 60). La qualità peggiora sicuramente quando non ci si basa sul genere oppure quando ci si basa su due generi comuni (profilo 6). Se prendiamo ad esempio il caso del profilo composto da film che accomunati dall'argomento trattato (profilo 4), o da film con due generi in comune(profilo 6), si può notare che spesso questi requisiti non sono rispettati. La qualità, valutata in base al fatto che vengano consigliati film di un genere simile ai film del profilo, è abbastanza positiva.

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni	Item-Item Cosine	KNN	Item-Item DR	Sarwar	Asymmetric SVD
1	-**L**: 5 -**FC :** Genere	MIN (Stesso genere)	Stesso genere(POP)	R: 60% (SI)	R:40% (SI)	R: 20% (NO)	R: 20% (SI)	R: 50% (NO)
		MIN (Stesso genere)	Altro genere(NOPOP)	R: 30% (SI)	R: 70% (SI, però forte presenza di altri generi)	R: 0% (NO)	R: 10% (SI, però forte presenza di altri generi)	R: 50% (NO)
2	-**L**: 3 -**Profilo non popolare** -**FC**: Attore , Genere	MIN (Stesso genere)	Stesso genere e attore (NOPOP)	R: 80% (SI)	R: 80% (SI)	R: 60% (SI, però forte presenza di altri generi)	R: 70% (SI)	R: 40% (NO)
		MIN (Stesso genere)	Stesso genere(NOPOP)	R: 90% (NO)	R: 90% (NO)	R: 100% (NO)	R: 100% (NO)	R: 50% (NO)
		MIN-MED (Stesso genere)	Altro genere, altro attore (NOPOP)	R: 100% (NO)	R: 100% (NO)	R: 90% (NO)	R: 100% (NO)	R: 60% (NO)
3	-**L**: 7 -**FC**: Genere	MIN (Stesso genere)	Stesso genere (NOPOP)	R: 30% (SI, però forte presenza di altri generi)	R: 0% (SI)	R: 10% (NO)	R: 0% (SI)	R: 40% (NO)
		MIN (Stesso genere)	Altro genere (NOPOP)	R: 100% (NO)	R: 100% (NO)	R: 70% (NO)	R: 80% (NO)	R: 70% (NO)
4	-**L**: 4 -**FC**: Argomento trattato	MIN (Stesso argomento)	Stesso argomento (NOPOP)	R: 50% (NO)	R: 50% (NO)	R: 10% (NO)	R: 20% (NO)	R: 50% (NO)
		MIN (Stesso argomento)	Altro argomento (NOPOP)	R: 100% (NO)	R: 100% (NO)	R: 40% (NO)	R: 50% (NO)	R: 40% (NO)

Tabella 4.3: Dataset Netflix: risultati profili ad hoc (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni	Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
5	**-L**: 4 **-Profilo non popolare** **-FC**: Regista	MIN	Stesso regista (NOPOP)	R: 60%	R: 50%	R: 20%	R: 80%	R: 50%
		MIN	Altro regista (POP)	R: 80%	R: 100%	R: 60%	R: 40%	R: 50%
6	**-L**: 3 **-Profilo non popolare** **-FC**: Generi, sequel	MIN (Stessi generi)	Stesso sequel (NOPOP)	R: 80%	R: 70%	R: 20%	R: 70%	R: 70%
		MIN (Stessi generi)	Stesso generi, no sequel (NOPOP)	R: 90% (SI,forte presenza film di uno dei due generi)	R: 90% (SI,forte presenza film di uno dei due generi)	R: 30% (SI,forte presenza film di uno dei due generi)	R: 70% (SI,forte presenza film di uno dei due generi)	R: 40% (SI, film di entrambi i generi)
		MIN-MED (Stessi generi)	Altro genere, no sequel (NOPOP)	R: 80% (SI, film di entrambi i generi)	R: 90% (SI, film di entrambi i generi)	R: 30% (SI, film di entrambi i generi)	R: 80% (SI,forte presenza film di uno dei due generi)	R: 50% (SI, film di entrambi i generi)
7	**-L**: 5 **-FC**: Popolarità	MIN	(POP)	R: 50%	R: 50%	R: 20%	R: 50%	R: 30%
		Nessun cambiamento	(NOPOP)	R: 40%	R: 0%	R: 0%	R: 30%	R: 60%
8	**-L**: 5 **-Profilo non popolare** **-FC**: Non popolarità	Nessun cambiamento	(NOPOP)	R: 80%	R: 100%	R: 20%	R: 20%	R: 50%
		MED	(POP)	R: 100%	R: 100%	R: 90%	R: 90%	R: 60%

Tabella 4.4: Dataset Netflix: risultati profili ad hoc (in grigio le aspettative non verificate)

Test sui profili ad hoc di Netflix: discussione

L'esito dei precedenti test, mostrato nelle Tabelle 4.3 e 4.4, ha sottolineato una notevole presenza di Reshuffling in tutte le varie fasi e in tutti gli algoritmi, con valori molto alti, fino a dei completi cambiamenti della lista di raccomandazione.

L'unico algoritmo che sembra comportarsi meglio, in pochissimi casi, è l'Item-Item DR, che basa le raccomandazioni soprattutto sui film con molte relazioni con altri film, viceversa il peggior algoritmo è risultato l'Item-Item Cosine, il quale presenta sempre un valore di Reshuffling più alto delle aspettative. Le previsioni di Reshuffling non sono state quindi assolutamente rispettate. Rispetto al dataset precedente, probabilmente, il fatto che questo sia molto più grande in termini di utenti e di film porta ad un degrado notevole della raccomandazione. I profili di partenza, inoltre, sono praticamente tutti popolari, ossia contengono almeno un film popolare, e in teoria proprio questi film, in algoritmi Collaborativi, dovrebbero influire maggiormente sulla raccomandazione(dovrebbero quindi limitare l'effetto di Reshuffling dovuto all'inserimento di un nuovo film), ma ciò non accade, sia nel caso si aggiungano film popolari, sia nel caso si aggiungano film non popolari.

Anche le previsioni sulla qualità della raccomandazione vengono parzialmente verificate (21 casi su 44) ma sicuramente il numero di previsioni esatte è maggiore rispetto a quello relativo alle previsioni sul valore di Reshuffling. La qualità peggiora sicuramente quando non ci si basa sul genere. Se prendiamo ad esempio il caso del profilo composto da film che trattano lo stesso argomento(profilo 4), o da film derivanti da un genere non famoso(profilo 6) noteremo che spesso questi requisiti non sono rispettati. La qualità valutata, in base al fatto che vengano consigliati film di un genere simile ai film del profilo, infatti, è abbastanza positiva.

Capitolo 5

Framework Antireshuffling

In questo capitolo verrà presentata la metodologia di risoluzione utilizzata per evitare l'eccessiva presenza di Reshuffling. Si tratta di un algoritmo correttivo, da applicare in coda ad un algoritmo di raccomandazione in diversi istanti di tempo a discrezione del gestore di sistema o del singolo utente.

5.1 Metodologia di risoluzione

L'algoritmo correttivo interviene su una nuova raccomandazione basandosi sulla raccomandazione precedente e andando a modificare il contenuto della nuova lista di raccomandazione. Il primo passo consiste nell'andare ad osservare quanti (e quali) elementi cambiano tra due liste di raccomandazione ottenute in due istanti di tempo diversi t_1 e t_2 con $t_1 < t_2$. Dal confronto di queste due liste (che chiameremo L_1 e L_2) vengono prodotte altre due liste, eventualmente vuote, chiamate *lista entranti* e *lista uscenti* così definite:

$$\text{lista entranti} = L2 \setminus L1 = \{x \in L2 \mid x \notin L1\} \tag{5.1}$$

$$\text{lista uscenti} = L1 \setminus L2 = \{x \in L1 \mid x \notin L2\} \tag{5.2}$$

Come suggerisce il nome, la prima lista corrisponde all'insieme di film che non erano presenti nella lista L_1 e che, in seguito ad una nuova richiesta di raccomandazione, sono presenti nella lista L_2. La seconda lista corrisponde, invece, all'insieme di film che erano presenti nella lista L_1 e che non sono più presenti nella lista L_2. Come si può notare, nel caso in cui non vi sia alcuna differenza tra le due liste di raccomandazione L_1 e L_2, *lista entranti* e *lista uscenti* saranno vuote. Una volta effettuata questa operazione, l'algoritmo si occupa di assegnare ad ogni film presente nelle due liste (entranti

e uscenti) una sorta di punteggio. Questo punteggio è assegnato in base a delle opportune metriche di valutazione che verranno trattate in seguito. Rifacendoci ancora alle due liste L_1 e L_2, l'idea di fondo è che in base a questo punteggio viene deciso se mantenere un film nella nuova lista di raccomandazione L_2 oppure se è il caso di mantenere un altro film della lista di raccomandazione precedente L_1 (ad esempio un film che nella nuova lista di raccomandazione è in terza posizione e che, nella lista precedente, era in trecentesima posizione ha avuto un incremento tale da far sì che possa essere mantenuto nella nuova lista di raccomandazione). Ovviamente questa decisione è limitata ai componenti della lista entranti e della lista uscenti. I film della lista entranti vengono ordinati con ordine decrescente in base al punteggio ottenuto, quelli della lista uscenti in ordine crescente. Una volta assegnati tutti i punteggi e una volta effettuato l'ordinamento si può passare alla terza fase, ossia quella relativa agli scambi. Innanzitutto la terza fase prevede che a priori sia stata stabilita una soglia (relativa ovviamente alla metrica di valutazione utilizzata) in base alla quale verranno effettuati gli scambi. All'interno di quest'ultima fase possiamo individuare due sottofasi:

- La prima sottofase prevede, in primis, un'analisi della lista entranti: tutti i film che hanno un punteggio superiore alla soglia vengono eliminati da tale lista e verranno mantenuti nella nuova lista di raccomandazione. Ogni volta che viene eliminato un film dalla lista entranti viene anche eliminato il film con punteggio più basso dalla lista uscenti. In questa maniera vengono premiati i film con un punteggio degno di nota. Il secondo passo consiste nell'analizzare la lista uscenti e eliminare, da tale lista, i film che hanno un punteggio minore della soglia negativa ($< -soglia$). Ogni volta che viene eliminato un film dalla lista uscenti viene anche eliminato il film con punteggio più alto dalla lista entranti. In questa maniera vengono penalizzati i film con punteggi molto bassi. Si noti che quest'ultima tipologia di confronto dipende dal tipo di metrica di valutazione utilizzata.

- La seconda sottofase prevede, invece, di valutare a coppie un film della lista entranti con un film della lista uscenti. Partendo dall'inizio delle due liste viene preso il punteggio del primo film della lista entranti e il punteggio del primo film della lista uscenti. Se la somma (spesso non è una somma, questo calcolo dipende comunque fortemente dalla metrica adottata) di tali punteggi supera la metà della soglia allora essi vengono eliminati dalle relative liste. In questa maniera vengono premiati i film che, a fronte di una nuova raccomandazione, hanno ottenuto un buon punteggio (non talmente alto da far sì che si superi

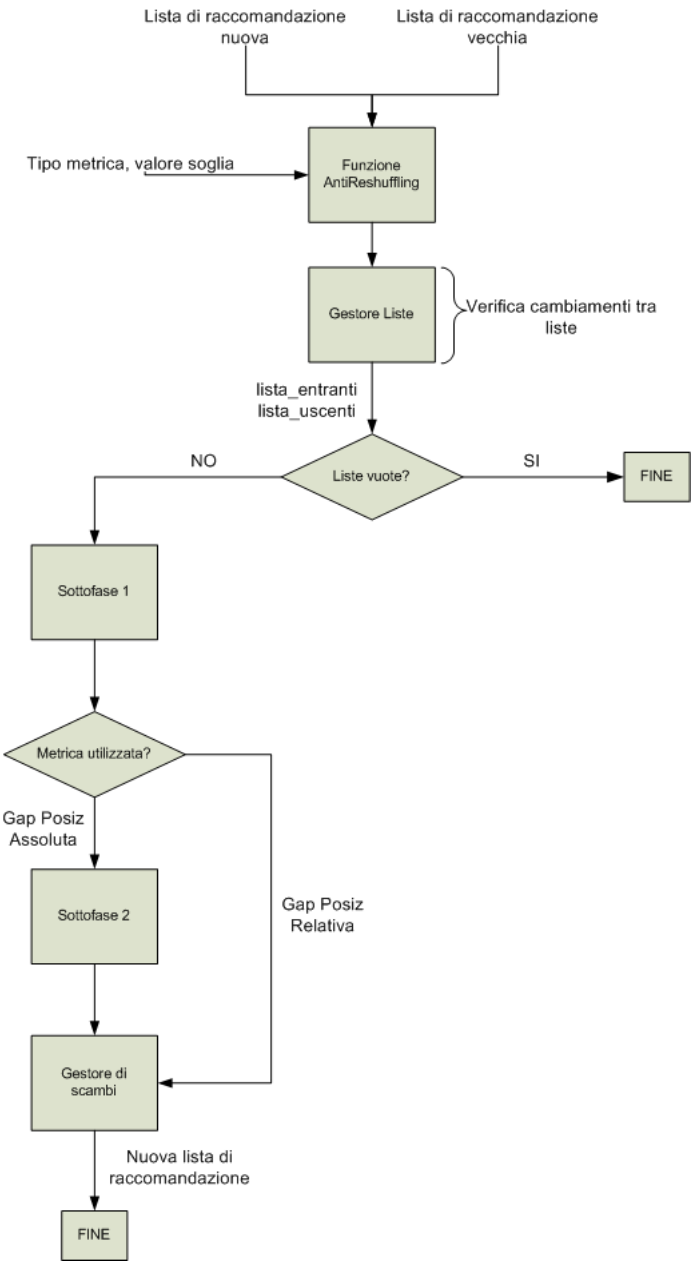


Figura 5.1: Workflow di funzionamento dell'algoritmo correttivo

la soglia) in corrispondenza di film che hanno ottenuto, uscendo dalla lista di raccomandazione, un punteggio basso o negativo. Questa sottofase non è sempre utilizzata. Come vedremo, infatti, il calcolo di questo divario non ha sempre senso. Per ogni metrica utilizzata verrà indicato se viene utilizzato il secondo metodo oppure no.

A questo punto, tutti i film presenti nella lista entranti corrispondono a film che ,secondo la normale raccomandazione, sarebbero stati raccomandati ma che non hanno ricevuto un punteggio tale da far sì che fossero meritevoli di rimanerci. Presa quindi, la nuova lista di raccomandazione, vengono presi i film che sono anche presenti nella lista entranti e vengono scambiati con i rimanenti nella lista uscenti. E' importante notare il fatto che le metriche di valutazione saranno comunque relative al comportamento del film in questione, come ad esempio l'aumento (o la diminuzione) di posizioni in classifica, l'aumento o diminuzione del punteggio assegnato dall'algoritmo etc. Nel seguente paragrafo verranno presentate due metriche che sono state utilizzate durante questo lavoro di tesi.

5.2 Metriche di valutazione

Le metriche di valutazione utilizzate sono relative all'aumento (o diminuzione) di posizioni che un determinato film ha ottenuto grazie alla nuova raccomandazione. Queste due metriche si differenziano per il modo in cui viene valutato questo aumento: in maniera assoluta o relativa alla posizione iniziale del film stesso.

5.2.1 Gap Posizione Assoluta

Questo tipo di metrica di valutazione va a valutare l'aumento/diminuzione di posizioni (in termini percentuali) di un determinato film. Essenzialmente, un algoritmo di raccomandazione, in seguito ad una nuova richiesta di raccomandazione, assegna ad ogni singolo film del sistema un determinato punteggio (in base ovviamente agli iPNut che ha ricevuto) ed ha come output la classifica di tutti i film forniti dal sistema, ordinati secondo il punteggio. Prendendo i primi N elementi della lista si ottiene la classifica TOP-N che sarà la vera e propria lista di raccomandazione presentata all'utente. Ogni singolo film fornito dal sistema, di conseguenza, avrà una determinata posizione in classifica. Prendendo in considerazione due raccomandazioni effettuate in tempi diversi, e considerando un singolo film, tale film avrà, dalla prima alla seconda raccomandazione, un cambiamento della propria

posizione in classifica, che potrà essere un aumento di posizione (aumenta il suo punteggio e di conseguenza il film scala la classifica) o una diminuzione di posizione. Questa metrica misura in termini percentuali questo aumento o diminuzione di posizione (in termini percentuali significa che l'aumento o la diminuzione vengono rapportati al numero di film forniti dal sistema). Teoricamente possiamo identificarla come segue:

$$\text{Gap} = \frac{PV - PN}{\#\text{film}} \tag{5.3}$$

Dove PV (PosizioneVecchia) indica la posizione in classifica del film nella vecchia lista di raccomandazione e PN (PosizioneNuova) la nuova posizione in classifica in seguito alla nuova raccomandazione. Come si può osservare, nel caso in cui ci sia un aumento di posizione in classifica il *gap* sarà positivo PN sarà minore di PV, in quanto il film sarà più in alto in classifica), nel caso in cui, invece, ci sia una diminuzione di posizione in classifica il *gap* sarà negativo (PN sarà maggiore di PV).

5.2.2 Gap Posizione Percentuale

Questo tipo di metrica di valutazione va a valutare l'aumento/diminuzione -relativo- di posizioni (in termini percentuali) di un determinato film. E' una metrica molto simile alla precedente con la piccola differenza che non ci si riferisce al numero di film totale ma è dipendente soltanto dalla posizione precedente e attuale del singolo film. Teoricamente possiamo identificarla come segue:

$$\text{Gap} = \frac{\text{PV} - \text{PN}}{\text{PV}} \text{se PN} > \text{PN} \tag{5.4}$$

$$\text{Gap} = \frac{\text{PN} - \text{PV}}{\text{PN}} \text{se PN} > \text{PV} \tag{5.5}$$

Viene utilizzato come denominatore la posizione maggiore tra PV e PN. Prendendo in considerazione due situazioni:

- Il film f_1 si trova al tempo t_1 in seconda posizione nella lista di raccomandazione ottenuta sempre al tempo t_1. Al tempo t_2 viene effettuata una nuova raccomandazione e il film si ritrova in ventesima posizione. In questo caso PN è maggiore di PV, quindi bisognerà utilizzare la seconda formula. Il *gap* ottenuto è pari a -0.9 e indica un decremento relativo del 90%.

- Il film f_1 si trova al tempo t_1 in ventesima posizione nella lista di raccomandazione ottenuta sempre al tempo t_1. Al tempo t_2 viene effettuata una nuova raccomandazione e il film si ritrova in seconda posizione. In questo caso PN è minore di PV, quindi bisognerà utilizzare la prima formula. Il *gap* ottenuto è pari a $+0.9$ ed indica un incremento relativo del 90%.

In questa maniera possiamo identificare con lo stesso valore un decremento e un incremento riferiti alle stesse posizioni (ovviamente scambiate tra un caso e l'altro). Utilizzando questa metrica non ha senso effettuare il secondo passo della fase di scambi proprio perchè questo *gap*, essendo relativo e calcolato per ogni film, si riferisce a posizioni diverse, e diventa quindi insensato andare a valutare a coppie i *gap* dei diversi film. Questo passo verrà quindi saltato nel caso si decida di utilizzare questa metrica.

5.3 Esempi di funzionamento

In questo paragrafo verrà presentato un tipico esempio di funzionamento dell'algoritmo correttivo. Per semplicità l'esempio sarà relativo a una raccomandazione Top-5 ossia l'output prodotto sarà una lista di raccomandazione di cinque elementi.

5.3.1 Gap Posizione Assoluta

Supponiamo di aver ottenuto al tempo t_1 una lista di raccomandazione L_1 e al tempo t_2 una lista di raccomandazione L_2, con $t_1 < t_2$. Si presuppone ovviamente che tra t_1 e t_2 il profilo utente dell'utente in questione sia stato modificato, ossia che l'utente abbia inserito dei nuovi voti o che comunque il modello su cui si basa la raccomandazione sia cambiato.

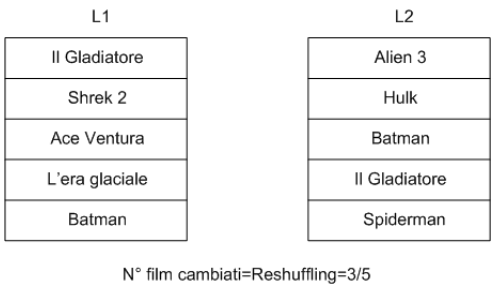


Figura 5.2: Gap Posizione Assoluta - Passo 1*: Lista al tempo* $t1$ *(sinistra) e lista al tempo* $t2$*(destra)*

Come si può vedere dalla Figura 5.2, queste due liste si differenziano per tre film: *Alien 3*, *Hulk* e *Spiderman* appaiono nella lista L_2 a discapito di *Shrek 2*, *Ace Ventura* e *L'era glaciale*. Il Reshuffling è quindi di $\frac{3}{5}$ pari quindi al 60%. Il secondo passo è quello di generare la lista entranti e la lista uscenti.

Lista entranti
Alien 3
Hulk
Spiderman

Lista uscenti
Shrek 2
Ace Ventura
L'era glaciale

Figura 5.3: Gap Posizione Assoluta - Passo 2*: Lista entranti (sinistra) e lista uscenti* $t2$*(destra)*

Come si può vedere dalla Figura 5.3 entrambe sono composte da tre film, la prima dai tre film che entrano nella nuova lista di raccomandazione, la seconda dai tre film che escono dalla vecchia lista di raccomandazione. Il passo successivo consiste nell'andare a calcolare per ogni singolo film di queste due liste il *Gap Posizione Assoluto* e ordinare la lista entranti secondo in maniera decrescente e la lista uscenti in quella crescente.

Lista entranti	GAP	P.I.	P.F.
Alien 3	20%	198°	1°
Hulk	10%	100°	2°
Spiderman	3%	35°	5°

Lista uscenti	GAP	P.I.	P.F.
Ace Ventura	-15%	2°	150°
L'era glaciale	-6%	3°	62°
Shrek 2	-1%	4°	14°

Soglia: 15% di 985 film = 148 film

Figura 5.4: Gap Posizione Assoluta - Passo 3*: Lista entranti (sinistra) e lista uscenti* $t2$*(destra) con relativo gap*

Dalla Figura 5.4 possiamo vedere ogni singolo valore assegnato ad ogni film delle due liste. Il valore è calcolato in base alla posizione del film nella vecchia lista di raccomandazione (colonna P.I.) e nella nuova lista (colonna P.F.). Possiamo anche notare il valore della soglia prefissata (15%) che sarà il valore in base al quale si effettueranno le cancellazioni dalle due liste. Il passo successivo infatti è quello di verificare, all'interno della lista entranti, se ci sono film il cui GAP Assoluto supera la soglia. Come si può notare ce n'è uno (*Alien 3*). Tale film verrà cancellato da questa lista, e verrà cancellato anche il primo della lista uscenti. Il risultato di questa operazione è presentato nella Figura 5.5.

Il passo successivo consiste nell'andare a verificare che nella lista uscenti vi siano dei film il cui gap sia minore della soglia negativa (-15%). In

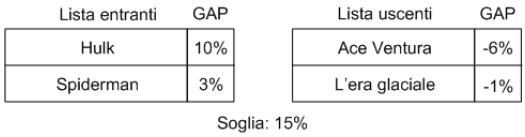


Figura 5.5: Gap Posizione Assoluta - Passo 4: Lista entranti (sinistra) e lista uscenti t2(destra) dopo il primo filtraggio

questo caso non c'è alcun film che abbia questa caratteristica, di conseguenza le liste rimangono così come sono. Il passo successivo consiste nel confronto a coppie. Si confronta il primo film della lista entranti con il primo film della lista uscenti. Se la differenza tra i due gap è maggiore della metà della soglia allora i due film vengono eliminati dalle rispettive liste. Si procede poi a coppie fino alla fine della lista. Ovviamente questo passo viene effettuato perchè si utilizza la metrica di valutazione *Gap Posizione Assoluta*. Partendo dall'inizio effettuiamo la prima operazione:

$$10\% - (-6\%) = 16\%$$

16% è maggiore della metà della soglia (ossia 7.5%), di conseguenza i film vengono eliminati dalle rispettive liste. Per gli altri due film invece 4% non è maggiore di 7.5%, quindi rimarranno all'interno di tali liste.

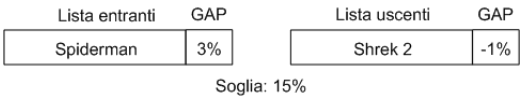


Figura 5.6: Gap Posizione Assoluta - Passo 5: Lista entranti (sinistra) e lista uscenti t2(destra) dopo il secondo filtraggio

Dalla Figura 5.6 si possono osservare i film rimasti nella lista entranti e in quella lista uscenti. Il film *Spiderman* non ha avuto un incremento di posizione tale da meritare di rimanere nella lista di raccomandazione (si noti che questo è il primo difetto di questa metrica che penalizza i film che sono eccessivamente vicini alle N posizioni della Top-N e che quindi non potranno avere un incremento tale da superare la soglia). I film verranno, di conseguenza, scambiati, ossia *Spiderman* verrà tolto dalla nuova lista di raccomandazione e verrà reinserito *Shrek 2*. La nuova lista di raccomandazione è presentata nella Figura 5.7.

Il nuovo valore di Reshuffling passa da $\frac{3}{5} = 60\%$ a $\frac{2}{5} = 40\%$.

Figura 5.7: Gap Posizione Assoluta - Passo 6*: Lista di raccomandazione restituita dall'algoritmo correttivo*

5.3.2 Gap Posizione Percentuale

Supponiamo di aver ottenuto al tempo t_1 una lista di raccomandazione L_1 e al tempo t_2 una lista di raccomandazione L_2, con t_1 ¡ t_2. Si presuppone ovviamente che tra t_1 e t_2 il profilo utente dell'utente in questione sia stato modificato, ossia che l'utente abbia inserito dei nuovi voti o che comunque il modello su cui si basa la raccomandazione sia cambiato.

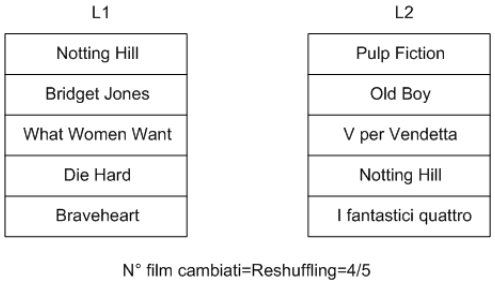


Figura 5.8: Gap Posizione Percentuale - Passo 1*: Lista al tempo* $t1$ *(sinistra) e lista al tempo* $t2$*(destra)*

Come si può vedere dalla Figura 5.8, queste due liste si differenziano per 4 film: *Pulp Fiction*, *Old Boy*, *V per Vendetta* e *I fantastici quattro* appaiono nella lista L_2 a discapito di *Bridget Jones*, *What Women Want*, *Die Hard* e *Braveheart*. Il Reshuffling è quindi di $\frac{4}{5}$ pari quindi all' 80%. Il secondo passo è quello di generare la lista entranti e la lista uscenti.

Come si può vedere dalla Figura 5.9 entrambe sono composte da 4 film, la prima dai 4 film che entrano nella nuova lista di raccomandazione, la seconda dai 4 film che escono dalla vecchia lista di raccomandazione. Il passo successivo consiste nell'andare a calcolare per ogni singolo film di queste due liste il *Gap Posizione Percentuale* e ordinare la lista entranti secondo l'ordine decrescente e la lista uscenti secondo l'ordine crescente.

Dalla Figura 5.10 possiamo vedere ogni singolo valore assegnato ad ogni film delle due liste. Il valore è calcolato in base alla posizione del film nella

Lista entranti
Pulp Fiction
Old Boy
V per vendetta
I fantastici quattro

Lista uscenti
Bridget Jones
What Women Want
Die Hard
Braveheart

Figura 5.9: Gap Posizione Percentuale - Passo 2: Lista entranti (sinistra) e lista uscenti t2(destra)

Lista entranti	GAP	P.I.	P.F.
Pulp Fiction	95%	20°	1°
Old Boy	91%	22°	2°
V per Vendetta	80%	15°	3°
I fantastici quattro	47%	9°	5°

Lista uscenti	GAP	P.I.	P.F.
Bridget Jones	-97%	2°	67°
What Women Want	-95%	3°	60°
Die Hard	-92%	4°	50°
Braveheart	-50%	5°	10°

Soglia: 90%

Figura 5.10: Gap Posizione Percentuale - Passo 3: Lista entranti (sinistra) e lista uscenti t2(destra) con relativo gap

vecchia lista di raccomandazione(colonna P.I.) e nella nuova lista (colonna P.F.). Possiamo anche notare il valore della soglia prefissata (90%) che sarà il valore in base al quale si effettueranno le cancellazioni dalle due liste. Il passo successivo infatti è quello di verificare, all'interno della lista entranti, se ci sono film il cui *Gap Posizione Percentuale* supera la soglia. Come si può notare ce ne son due (*Pulp Fiction* e *Old Boy*). Tali film verranno cancellati da questa lista, e verranno cancellati anche i primi due della lista uscenti. Il risultato di questa operazione è presentato nella Figura 5.11.

Lista entranti	GAP
V per Vendetta	80%
I fantastici quattro	47%

Lista uscenti	GAP
Die Hard	-92%
Braveheart	-50%

Soglia: 90%

Figura 5.11: Gap Posizione Percentuale - Passo 4: Lista entranti (sinistra) e lista uscenti t2(destra) dopo il primo filtraggio

Il passo successivo consiste nell'andare a verificare che nella lista uscenti vi siano dei film il cui gap sia minore della soglia negativa (-15%). Vi è un film che ha questa caratteristica, ossia *Die Hard*, che verrà cancellato dalla lista uscenti. Verrà anche cancellato il rispettivo della lista entranti. Come già detto, con questa metrica il confronto a coppie è completamente insensato. Il procedimento di analisi delle liste entranti e uscenti è terminato e si può procedere con gli scambi.

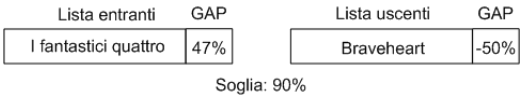


Figura 5.12: Gap Posizione Percentuale - Passo 5*: Lista entranti (sinistra) e lista uscenti* $t2$*(destra) dopo il secondo filtraggio*

Come si nota dalla Figura 5.12 questi sono i film rimasti nella lista entranti e lista uscenti. Il film *I fantastici Quattro* non ha avuto un incremento di posizione relativa tale da meritare di rimanere nella lista di raccomandazione. I film verranno, di conseguenza, scambiati, ossia *I fantastici quattro* verrà tolto dalla nuova lista di raccomandazione e verrà reinserito *Braveheart*. La nuova lista di raccomandazione è presentata nella Figura 5.13.

Figura 5.13: Gap Posizione Percentuale - Passo 6*: Lista di raccomandazione restituita dall'algoritmo correttivo*

Il nuovo valore di Reshuffling passa da $\frac{4}{5} = 80\%$ a $\frac{3}{5} = 60\%$.

Capitolo 6

Metodologia di test e risultati

In questa sezione vengono descritte le metodologie di test per l'algoritmo sviluppato, le metriche di qualità considerate per la valutazione di quest'ultimo e infine vengono presentati i risultati ottenuti anche in riferimento alla valutazione del problema discussa nel capitolo 3.

6.1 K-Fold Cross Validation

La K-Fold Cross Validation è una tecnica che permette di stimare quanto possono essere affidabili i risultati di un'analisi statistica su un campione indipendente [17]. Partendo da un campione, o *dataset*, questo viene suddiviso casualmente in K parti, o sottocampioni, della stessa numerosità. La K-esime parte viene chiamata test set, o campione di validazione, mentre i restanti $K-1$ sottocampioni vengono definiti training set. Il procedimento viene ripetuto K volte in modo che ogni sottocampione possa far parte del test set esattamente una sola volta. Durante ogni campionamento il training set viene utilizzato per la costruzione del modello, mentre il test set viene utilizzato per la validazione di questo modello. I risultati ottenuti dopo i K campionamenti possono essere combinati per ottenere una singola stima. Il vantaggio di questo metodo è che ogni singolo elemento viene usato almeno una volta per la validazione del modello in modo da evitare problemi di overfitting. Solitamente K viene settato a 10 e anche in questo lavoro si è deciso di utilizzare il metodo di 10-fold cross validation.

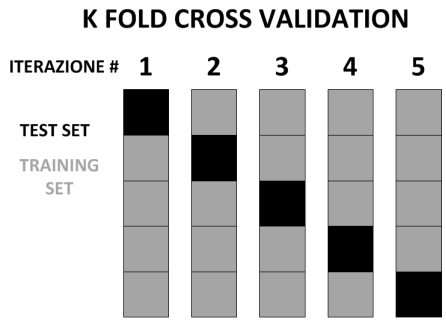


Figura 6.1: Esempio metodo K-Fold Cross Validation

6.2 Bootstrap

Il bootstrap è una tecnica statistica di ricampionamento usata per approssimare la distribuzione campionaria, quando questa è sconosciuta, di un parametro statistico p , in modo da poterne calcolare media e varianza e successivamente il relativo intervallo di confidenza. Partendo da un campione osservato X di numerosità n composto quindi dagli elementi $(x_1, ..., x_n)$, questo viene campionato m volte in modo da ottenere m campioni $(Y_1, ..., Y_n)$ di numerosità n. Ogni elemento del campione X può venire estratto più di una volta in ogni ciclo di bootstrap, ogni volta con probabilità $frac1n$. Per esempio se partendo da un campione $X = 1, 2, 3, 4, 5$ al primo ciclo di bootstrap posso otteneere un campione $Y_1 = 5, 4, 2, 2, 1$. Per ogni campione Y, ottenuto in tutti i cicli di bootstrap, si va a misurare il parametro statistico p preso in considerazione, in modo da ottenere m stime di p così da poterne calcolare media e varianza. In questo lavoro si è scelto di utilizzare $m = 100$.

6.3 Metriche di qualità

Le metriche di qualità si suddividono in due categorie :

- Metriche di errore
- Metriche di classificazione

Tra le metriche di errore sono presenti MAE, MSE e RMSE [13]. MAE (Mean Absolute Errore lett. Errore Assoluto Medio) rappresenta la distanza media (in valore assoluto) tra il valore predetto e il valore reale. Viene calcolato tramite la seguente formula:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} \mid \hat{r}_{ij} - r_{ij} \mid \tag{6.1}$$

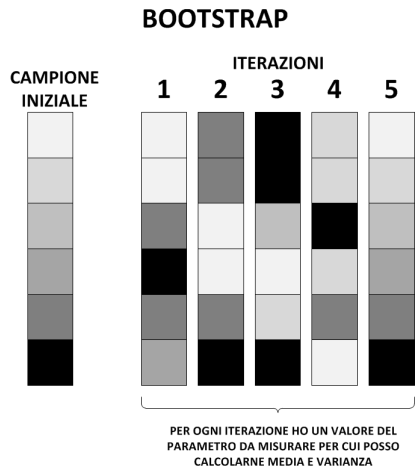


Figura 6.2: Esempio metodo Bootstrap

MSE (Mean Squared Error lett Errore Quadratico Medio) indica la differenza quadratica media trai valori osservati e i valori attesi. Viene calcolato tramite la seguente formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{r}_{ij} - r_{ij})^2 \tag{6.2}$$

RMSE (Root Mean Squared Error lett.Radice dell'Errore Quadratico Medio) corrisponde alla varianza interna data dal rapporto fra la devianza interna e la numerosità totale. Viene calcolato come:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{6.3}$$

Queste metriche di errore vengono utilizzate, nell'ambito dei sistemi di raccomandazione, per valutare la predizione di un rating, in particolare quelli espliciti. Dato che l'algoritmo correttivo non influisce in nessun modo sul rating predetto ma soltanto sulla posizione nell lista di raccomandazione, queste metriche non sono state utilizzate nel corso di questo lavoro. Al contrario , in questo lavoro, le metriche di classificazione, tra le quali le principali sono Recall e Precision [13] sono state utilizzate per valutare la bontà della raccomandazione dei diversi algoritmi con l'applicazione del metodo correttivo. Nelle sezioni successive vengono presentate le metriche utilizzate.

6.3.1 Recall

Ogni item che viene raccomandato dall'algoritmo può essere rilevante o non per un determinato utente. Un item rilevante è un item che piace sicuramente all'utente e che questi voterebbe con un rate alto. Un item non

rilevante al contrario è un item a cui l'utente non è interessato e voterebbe con un rate basso. Di conseguenza ogni item può essere classificato in una delle seguenti categorie :

VP (Veri Positivi) item consigliati dal sistema e rilevanti per l'utente.

FP (Falsi Positivi) item consigliati dal sistema e non rilevanti per l'utente.

VN (Veri Negativi) item non consigliati dal sistema e non rilevanti per l'utente.

FN (Falsi Negativi) item non consigliati dal sistema ma rilevanti per l'utente.

La recall è una delle metriche di qualità più accurata per un algoritmo di raccomandazione [22]. Questa metrica è calcolata come [4]:

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{6.4}$$

ovvero la capacità del sistema di raccomandare degli item davvero interessanti, intesa come percentuale degli item consigliati tra tutti quelli che risultano rilevanti per l'utente, per questo può essere anche vista come una metrica di completezza. Questa metrica dipende molto dalla lunghezza della Top-N list. In questo lavoro, preso un campione di raccomandazioni e considerando una Top-10 list la recall è stata calcolata come la percentuale degli item consigliati nelle prime dieci posizioni (quindi appartenenti alla lista di raccomandazione) rispetto alla lunghezza totale del campione, per questo valori più alti di recall indicano un algoritmo più accurato.

6.3.2 Precision

La precision è anch'essa una delle metriche di qualità più accurata per un algoritmo di raccomandazione [22]. E' calcolata come:

$$\text{Precision} = \frac{\text{VP}}{N} \tag{6.5}$$

A differenza della recall che è una misura di completezza, la precision è una misura di esattezza e affidabilità, infatti misura la percentuale di item consigliati rilevanti per l'utente rispetto al numero di quelli raccomandati e non rispetto al totale degli item. Questa metrica dipende anch'essa dalla lunghezza della Top-N.

6.3.3 EPR

L'Expect Percentile Ranking (lett. votazione attesa percentuale) è una misura di qualità che non considera soltanto gli item raccomandati nella TOP-N list. Preso infatti un campione di raccomandazioni l'EPR indica percentualmente la posizione in classifica in cui vengono consigliati gli item. Questa metrica è calcolata attraverso la seguente formula [14]:

$$\text{EPR} = \frac{\sum_{u,i} r_u^T i \times rank_u i}{\sum_{u,i} r_u^T i} \tag{6.6}$$

Valori di EPR più bassi sono preferibili perchè indicano che il sistema ha predetto gli item visti dall'utente nelle prime posizioni. Un valore del 50% è ottenibile per delle raccomandazioni casuali e dei valori superiori al 50% sono ancora peggiori. In questo lavoro preso un campione di raccomandazioni l'EPR è calcolato come la somma delle posizioni in classifica di tutti gli elementi del campione rispetto alla sua numerosità.

6.3.4 ARHR

L'Average Reciprocal Hit Rank è una misura di qualità che prende in considerazione la posizione in cui un item viene consigliato all'interno della lista di raccomandazione. Definiamo *hit* un item raccomandato, e quindi presente nella Top-N list. Preso un campione di raccomandazioni di numerosità n possiamo calcolare un parametro definito HR (Hit Rate) come :

$$\text{HR} = \frac{\#\text{Hits}}{n} \tag{6.7}$$

Un Hit Rate pari a 1 significa che l'algoritmo è stato in grado di consigliare tutti gli item. Questo parametro ha però lo svantaggio di non considerare la posizione in classifica in cui viene consigliato un determinato item, infatti non c'è differenza se tutti gli hit avvengono in prima posizione o tutti nell'ultima posizione della Top-N list. Questa limitazione può essere superata tramite la metrica dell'ARHR, che da un punteggio ad ogni hit in base alla posizione nella Top-N in cui avviene. Se h è il numero di hits che occorrono nelle posizioni $p_1, p_2, ..., p_N$, l'ARHR è calcolato come [10]:

$$\text{ARHR} = \frac{1}{n} \sum_{i=1}^{h} \frac{1}{p_i} \tag{6.8}$$

In questo modo gli hits che avvengono nelle prima posizioni della lista di raccomandazione sono pesati di più rispetto a quelli nelle ultime posizioni.

Quando tutti gli item vengono consigliati in prima posizione si ottengono i valori più alti di ARHR, il quale coincide con l'Hit Rate; valori bassi si ottengono invece quando le raccomandazioni avvengono nelle ultime posizioni della Top-N, al peggio l'ARHR è pari a $\frac{HR}{N}$.

6.4 Prima metodologia: test statistici

A differenza dei test per la valutazione della presenza di Reshuffling, in questa fase ci si è concentrati soltanto sui profili utente con al massimo dieci rating. Questo perchè, come è risultato nel capitolo 3, le prime due categorie sembrano essere le più affette da questo problema. Alla lunghezza massima del profilo devono essere aggiunti due rating che verranno utilizzati per la valutazione del test, per cui i profili ammissibili per questo test sono quelli di lunghezza compresa tra 3 rating e 12 rating. Il test è stato svolto tramite un metodo di K-Fold Cross Validation prendendo come parametro $K = 10$. Inizialmente si è riordinata casualmente la matrice URM, in modo che la presenza di un determinato utente in un certo fold fosse casuale, e quindi non ci fosse nessun tipo di legame tra gli utenti appartenenti ad uno stesso fold. Successivamente per le dieci iterazioni previste dal metodo di Cross Validation sono stati effettuati i seguenti passi:

- Con i nove fold facenti parte il *training set* si è costruito il modello m per ogni algoritmo testato.
- Per ogni profilo che rispetta le caratteristiche volute (3 < lunghezza <= 12) del fold considerato come *test set* sono state eseguite le seguenti azioni:
  - Viene tolto dal profilo un rating che verrà poi utilizzato per il calcolo della metriche di qualità,nel caso di rating espliciti, il rating verrà considerato solo se pari a 5, ovvero un rating positivo).
  - Viene tolto dal profilo un rating per la valutazione del Reshuffling,nel caso di dataset esplicito il rating verrà considerato solo se pari a 4 o a 5,cioè dei voti positivi. Inoltre rating con un valore minore potrebbero non influire sulla raccomandazione e quindi non generare Reshuffling.
  - Viene calcolata la lista di raccomandazione che chiamiamo $ListaBase$ per il profilo riscalato, dopodichè viene riaggiunto l'ultimo rating eliminato e ricalcolata una nuova lista di raccomandazione che definiamo $ListaNoAnti$.

Figura 6.3: Modifiche effettuate al profilo durante il test

- La differenza tra queste due liste viene analizzata dall'algoritmo correttivo il quale restituisce una nuova lista definita $ListaAnti$, riducendo, quando opportuno, il valore di Reshuffling.
- Per ogni raccomandazione vengono memorizzate la differenza tra la $ListaBase$ e le due liste $ListaNoAnti$ e $ListaAnti$ per la valutazione del Reshuffling e la posizione nelle due liste $ListaNoAnti$ e $ListaAnti$ del primo item tolto dal profilo per il calcolo delle metriche di qualità.

Tramite il campione di raccomandazioni posso calcolarmi il valore di Reshuffling e le diverse metriche di qualità sia applicando l'algoritmo correttivo che per la raccomandazione normale.

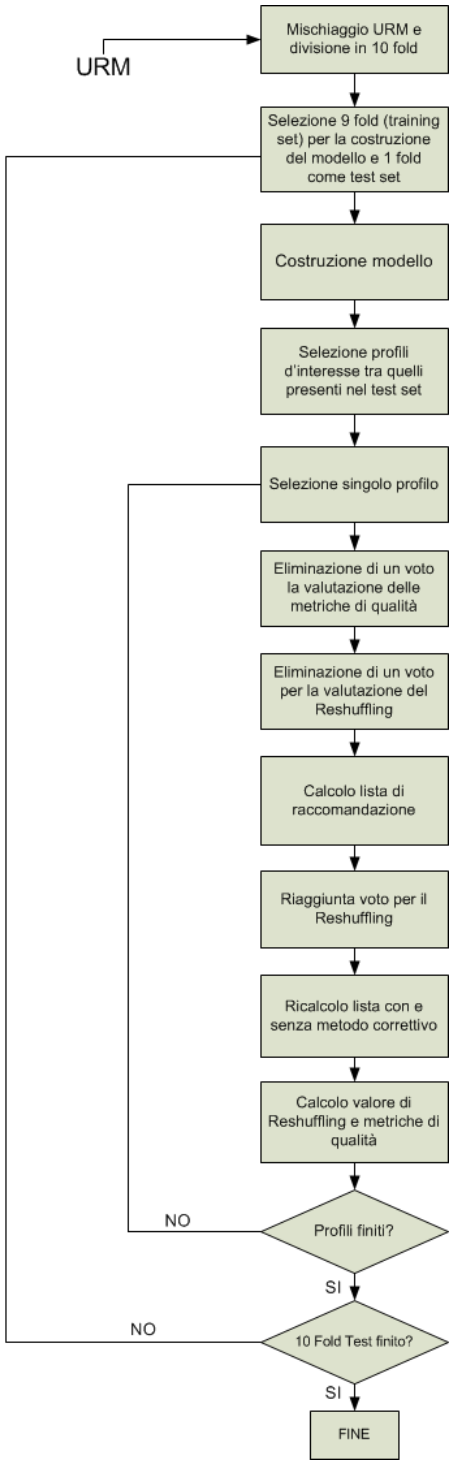


Figura 6.4: Worflow Test Statistico: Ten Fold Cross Validation

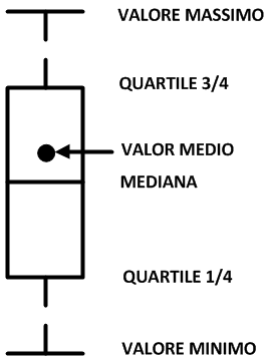
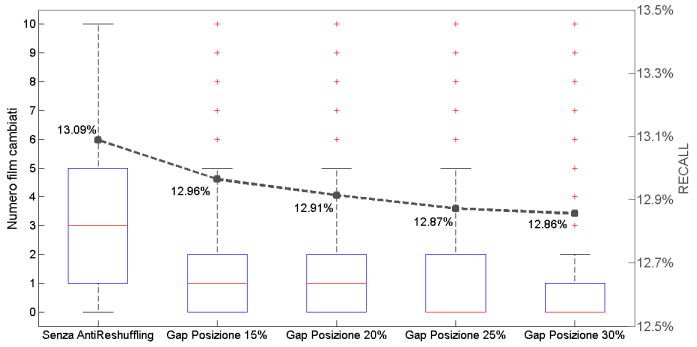


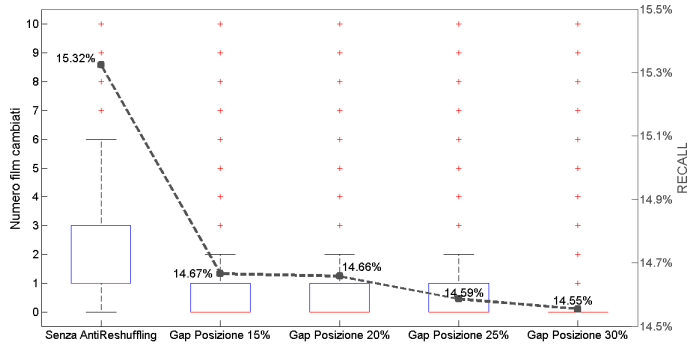
Figura 6.5: Caratteristiche Boxplot

6.5 Risultati test statistici

Per rappresentare i risultati della prima metodologia di test sono stati utilizzati dei grafici di tipo boxplot, i quali descrivono meglio la distribuzione di un campione attraverso indici di dispersione e di posizione. Il boxplot è rappresentato da un rettangolo delimitato dal primo e dal terzo quartile $q_{\frac{1}{4}}, q_{\frac{3}{4}}$ e una linea che rappresenta la mediana del campione. I segmenti uscenti dal rettangolo sono invece delimitati dal valore minore e maggiore appartenenti al campione. In questo modo vengono rappresentati graficamente i quattro intervalli ugualmente popolati delimitati dai quartili. Per aumentare l'affidabilità della misurazione della Recall è stato inoltre applicato un metodo di bootstrap in modo da poterne identificare un'intervallo di confidenza. Nei risultati l'intervallo non è riportato perchè nell'ordine di centesimi di percentuale (quindi di poca rilevanza).

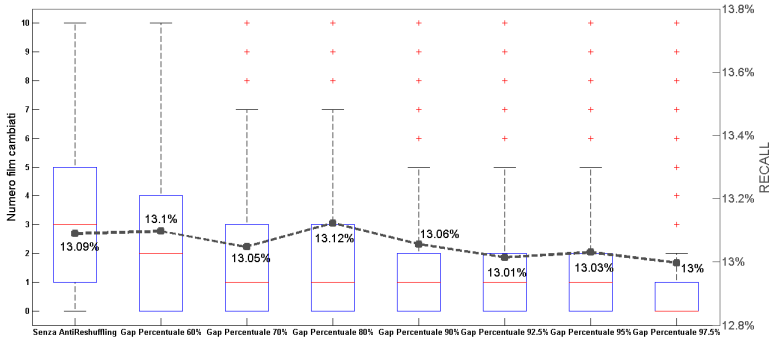


(a) Aggiunta film popolare

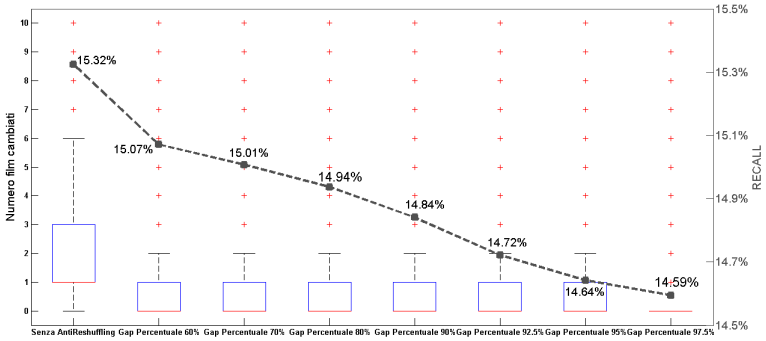


(b) Aggiunta film non popolare

Figura 6.6: Dataset MovieRec - Algoritmo Sarwar - Metrica Gap Posizione Assoluta: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b)

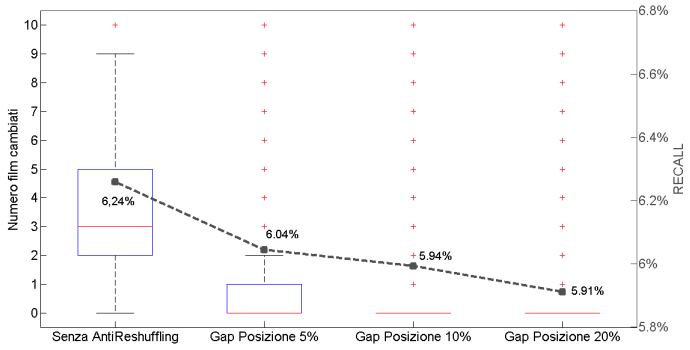


(a) Aggiunta film popolare

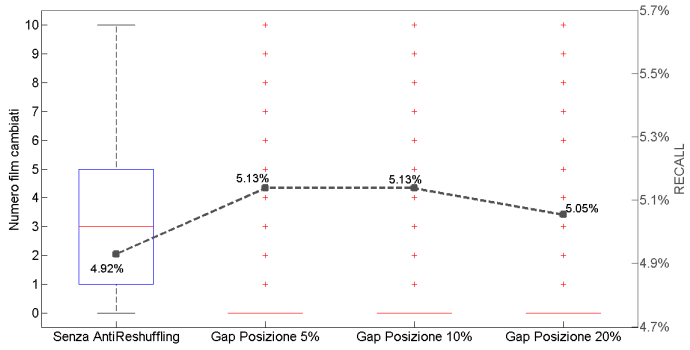


(b) Aggiunta film non popolare

Figura 6.7: Dataset MovieRec - Algoritmo Sarwar - Metrica Gap Posizione Percentuale: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b)

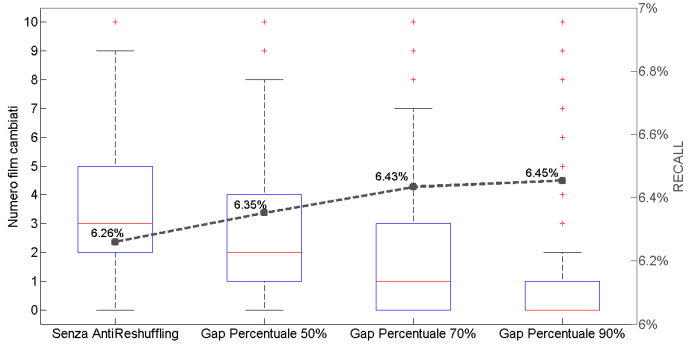


(a) Aggiunta film popolare

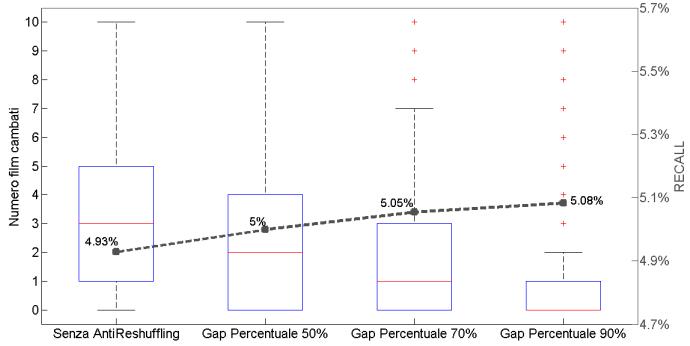


(b) Aggiunta film non popolare

Figura 6.8: Dataset Netflix - Algoritmo AsymmetricSVD - Metrica Gap Posizione Assoluta: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b)

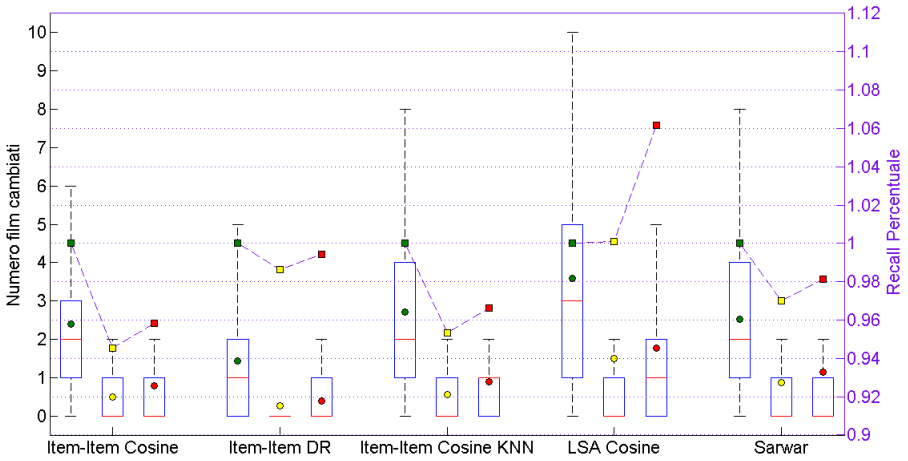


(a) Aggiunta film popolare

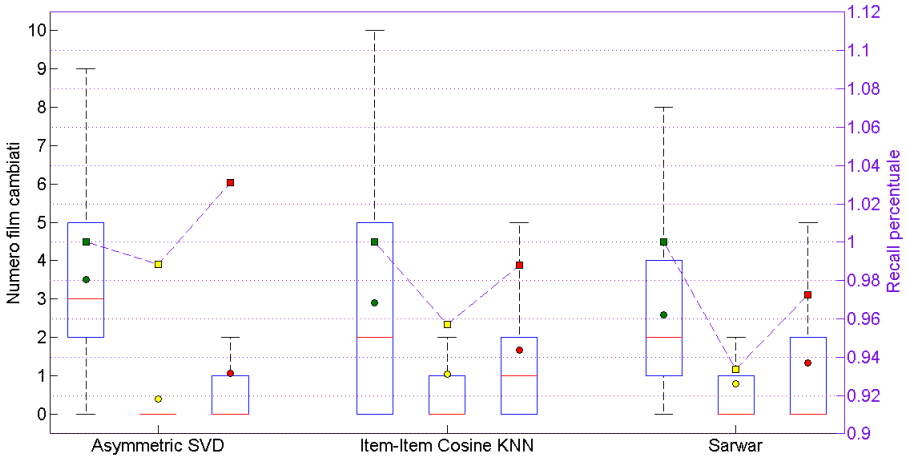


(b) Aggiunta film non popolare

Figura 6.9: Dataset Netflix - Algoritmo AsymmetricSVD - Metrica Gap Posizione Percentuale: Valori di Reshuffling e Recall aggiungendo un film popolare (a) o non popolare (b



(a) Dataset MovieRec



(b) Dataset Netflix

Figura 6.10: Grafici Riassuntivi con Confronto Metriche per il dataset MovieRec (a) e Netflix (b)

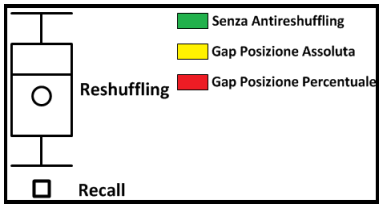


Figura 6.11: Legenda

Commenti sui risultati dei test statistici

In seguito ai primi risultati ottenuti dai test, in cui è stato applicato l'algoritmo di Antireshuffling, si è notato che il comportamento dei profili popolari e non, in seguito all'aggiunta di un film popolare o non, era più o meno simile o comunque la differenza di comportamento era irrilevante. Di conseguenza si è deciso di studiare i profili senza tener conto della loro popolarità, concentrandosi in particolare sul diverso effetto scaturito dall'aggiunta di film popolari o non. Il metodo correttivo è stato applicato per entrambe le metriche di valutazione con diverse soglie, in modo da poter meglio analizzare la presenza di Reshuffling nei vari casi e poter in qualche modo scegliere la metrica (e la rispettiva soglia) che fornisce il miglior compromesso tra valore di Reshuffling e metriche di qualità. I grafici è presentatano solo l'evoluzione della Recall (principale metrica di qualità considerata), mentre le tabelle presenti nelle pagine successive riportano i valori relativi alle altre metriche di qualità.

Le prime figure riportate (6.6(a), 6.6(b), 6.7(a), 6.7(b)) si riferiscono al comportamento dell'algoritmo Sarwar sul dataset MovieRec (rating impliciti). Da questi grafici si nota immediatamente la sensibile riduzione del valore di Reshuffling, soprattutto nel caso in cui viene utilizzata la metrica *Gap Posizione Assoluta*. Tale metrica è sicuramente più restrittiva rispetto all'altra metrica proprio perchè considera l'incremento di posizione in termini assoluti. Nel caso di metrica *Gap Posizione Percentuale* molti più film sono in grado di superare il confronto con la soglia prefissata e quindi la diminuzione di Reshuffling viene più attenuata.

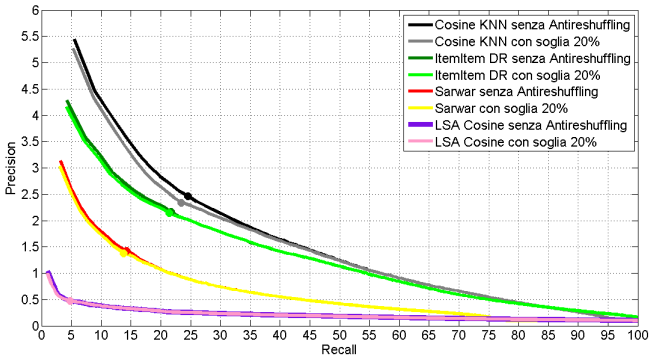
Analizzando l'effetto di film popolari o non, la prima considerazione che si può evincere è sicuramente che l'aggiunta di film popolari porta ad una maggiore presenza di Reshuffling, anche dopo l'applicazione dell'algoritmo correttivo, rispetto al caso non popolare. In entrambi ovviamente, l'algoritmo correttivo diminuisce, all'aumentare della soglia, il valore di Reshuffling. Nel caso di aggiunta di film non popolari, per i valori di soglia maggiori, l'intervento dell'algoritmo correttivo è talmente rilevante da far si che la presenza di Reshuffling sia pressochè nulla (salvo alcuni outlier). Nonostante l'intervento dell'algoritmo correttivo la Recall subisce una diminuzione quasi irrilevante (al più di un punto percentuale). Decresce maggiormente nel caso di aggiunta di film non popolari ma i risultati ottenuti rimangono comunque soddisfacenti perchè il livello di qualità ottenuto è pressoche lo stesso di quello ottenuto dall'algoritmo originale. Durante l'applicazione della metrica *Gap Posizione Percentuale* possiamo notare, in due casi particolari, come la Recall subisca un lieve aumento rispetto al caso originale.

Le successive figure riportate (6.8(a), 6.8(b), 6.9(a), 6.9(b)) si riferiscono al comportamento dell'algoritmo Asymmetric SVD sul dataset Netflix (rating espliciti). Da questi grafici, in primis, possiamo notare come l'aumento della Recall, che nel caso precedente era presente in casi eccezionali, sia una realtà ben definita. Soprattuto con l'utilizzo della metrica *Gap Posizione Percentuale*, nonostante il valore di Reshuffling diminuisca sensibilmente come da aspettative, la Recall subisce un incremento che cresce all'aumentare della soglia. Tale incremento rimane comunque nell'ordine dei decimi di percentuale.

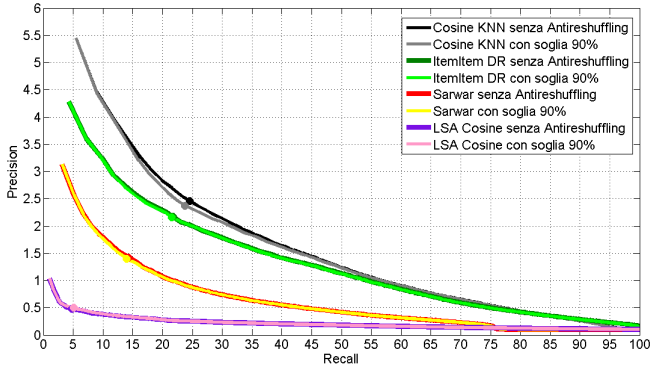
Per quanto riguarda l'utilizzo delle due metriche di valutazione possiamo notare che la metrica *Gap Posizione Assoluta* interviene significativamente indistintamente dalla popolarità del film aggiunto. Per ottenere valori di Reshuffling maggiori, in modo da avere una maggiore dinamicità della lista, si potrebbe diminuire la soglia, relativamente a questo algoritmo che sembra, a differenza degli altri algoritmi testati il pià influenzato da questo tipo di metrica di valutazione. I risultati ottenuti con la metrica *Gap Posizione Percentuale* sono soddisfacenti sia in termini di Reshuffling che di qualità della raccomandazione. Nelle Figure 6.10(a) e 6.10(b) sono mostrati i risultati ottenuti dai test per tutti gli algoritmi di raccomandazione utilizzati senza distinzione di popolarità del profilo o dei film aggiunti. In questo caso sono riportati i risultati originali e quelli ottenuti utilizzando l'algoritmo correttivo con entrambe le metriche e per ognuna rispettivamente il valore di soglia utilizzato nei test sui profili ad hoc. In questi grafici la Recall riferita alle due metriche correttive è una percentuale della Recall ottenuta dall'algoritmo di raccomandazone senza metodo correttivo.

I risultati mostrati in Figura 6.10(a), riferita al dataset MovieRec, possono essere considerati soddisfacenti, in quanto il Reshuffling diminuisce (viene quasi sempre dimezzato) in tutti gli algoritmi con una perdita di Recall inferiore al decimo di percentuale. Si può notare come la metrica *Gap Posizione Assoluta* porti a valori di Reshuffling inferiori rispetto all'altra metrica, ma in quest'ultima la Recall è leggermente più alta. Per l'algoritmo LSA Cosine abbiamo un aumento della Recall rispetto all'originale per entrambe le metriche. Per i risultati mostrati in Figura 6.10(b), riferita al dataset Netflix, valgono le stesse considerazioni del dataset MovieRec in particolare per quanto riguarda la Recall, nell'algoritmo Asymmetric SVD, con la metrica *Gap Posizione Percentuale* è presente un lieve aumento della Recall rispetto all'originale.

Nelle Figure 6.12(a), 6.12(b), 6.13(a) e 6.13(b) viene mostrato il rapporto tra le due metriche di qualità più importanti: la Precision (vedi Sezione 6.3.2) e la Recall (vedi Sezione 6.3.1). L'algoritmo correttivo è stato appli-

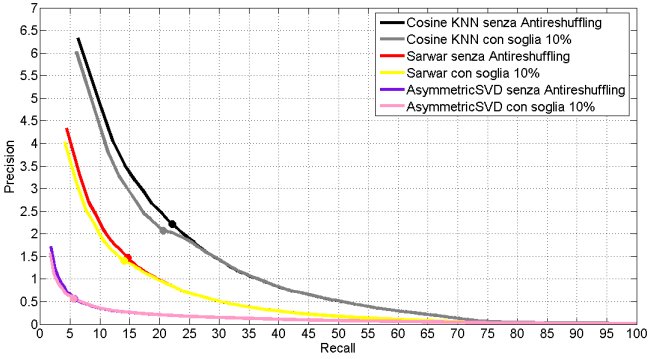


(a) Metrica Gap Posizione Assoluto

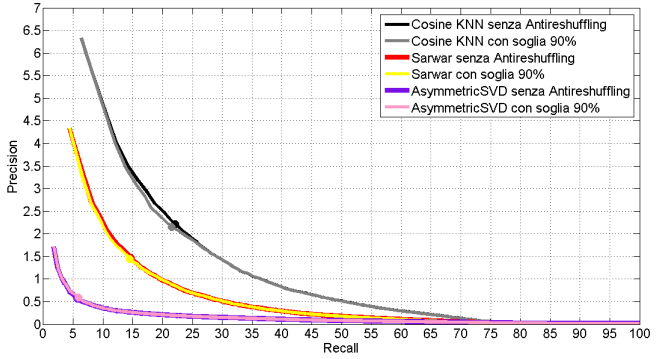


(b) Metrica Gap Posizione Percentuale

Figura 6.12: Dataset MovieRec - Recall / Precision per la metrica Gap Posizione Assoluta (a) e Percentuale (b) (L'indicatore è posto a N=10



(a) Metrica Gap Posizione Assoluto



(b) Metrica Gap Posizione Percentuale

Figura 6.13: Dataset Netflix - Recall / Precision per la metrica Gap Posizione Assoluta (a) e Percentuale (b) (L'indicatore è posto a N=10)

cato con entrambe le metriche con le soglie utilizzate nei test sui profili ad hoc contenuti nella sezione successiva. Si può notare come (considerando i valori per N=10 segnati con l'indicatore) la Precision si abbassa leggermente in relazione ad un abbassamento della Recall. Trai diversi algoritmi testati si può notare che l'Item-Item Cosine KNN con l'aggiunta del metodo correttivo ha la maggior diminuzione della Recall, negli algoritmi Item-Item DR e Sarwar invece i valori di Recall e Precision sono praticamente identici sia senza che con algoritmo correttivo. LSA e Asymmetric SVD invece, con il metodo correttivo, come già succedeva in precedenza hanno un lieve aumento della Recall.

6.5.1 Risultati riassuntivi dei test statistici

Tabella 6.1: Tabelle Riassuntive Metriche di Qualità per i dataset MovieRec(a) e Netflix(b)

(a) Dataset MovieRec

		Reshuffling	Recall	EPR	ARHR
Item-Item Cosine	**Originale**	2,4±0.02	24,18%	10,34%	10,21%
	Gap Posizione Assoluta	0,49±0.01	22,85%	10,35%	9,78%
	Gap Posizione Percentuale	0.79±0.04	23,17%	10,54%	10,04%
Item-Item Cosine KNN=100	**Originale**	2,71±0.03	24,58%	12,42%	10,26%
	Gap Posizione Assoluta	0,55±0.02	23,43%	12,44%	9,88%
	Gap Posizione Percentuale	0,89±0.02	23,75%	12,59%	10,11%
Item-Item DR	**Originale**	1,43±0.02	21,68%	10,65%	8,53%
	Gap Posizione Assoluta	0,26±0.01	21,38%	10,66%	8,35%
	Gap Posizione Percentuale	0,39±0.01	21,56%	10,83%	8,50%
Sarwar	**Originale**	2,53±0.03	14,23%	29,97%	5,29%
	Gap Posizione Assoluta	0,86±0.02	13,81%	29,98%	5,73%
	Gap Posizione Percentuale	1,13±0.02	13,97%	30%	5,85%
LSA Cosine	**Originale**	3,6±0.04	4,77%	35,6%	1,93%
	Gap Posizione Assoluta	1,5±0.04	4,78%	35,6%	1,88%
	Gap Posizione Percentuale	1,77±0.04	5,07%	35,6%	1,97%

(b) Dataset Netflix

		Reshuffling	Recall	EPR	ARHR
Item-Item Cosine KNN=100	**Originale**	2,9±0.04	22,1%	13,9%	10,53%
	Gap Posizione Assoluta	1,04±0.04	20,64%	13,91%	9,9%
	Gap Posizione Percentuale	1,66±0.03	21,5%	14,01%	10,38%
Sarwar	**Originale**	2,57±0.04	14,66%	26,42%	7,07%
	Gap Posizione Assoluta	0,78±0.02	14,03%	26,42%	6,64%
	Gap Posizione Percentuale	1,32±0.03	14,48%	26,43%	7,01%
Asymmetric SVD	**Originale**	3,5±0.04	5,7%	23,23%	2,73%
	Gap Posizione Assoluta	0,39±0.02	5,63%	23,23%	2,56%
	Gap Posizione Percentuale	1,06±0.03	5,87%	23,24%	2,75%

Dalle Tabelle 6.1(a) e 6.1(b) è possibile valutare per mezzo delle metriche di qualità i risultati ottenuti dall'algoritmo correttivo applicando le due metriche con valori di soglia del 20% per la *Gap Posizione Assoluta* e 90% per la *Gap Posizione Percentuale*. Ogni riga della tabella rappresenta i risultati ottenuti da ogni algoritmo testato, mentre ogni sottoriga è riferita al caso base o alle due metriche utilizzate. Ogni colonna, invece, rappresenta il valore di Reshuffling con il relativo intervallo di confidenza o il rispettivo valore della metrica di qualità (vedi Sezione 6.3.1, 6.3.3 e 6.3.4). In tutti gli algoritmi si può notare un abbassamento del Reshuffling almeno della metà rispetto al valore originale. Le metriche di qualità, a parte la Recall, subiscono dei peggioramenti di poca entità e quindi praticamente irrilevanti. Nella Tabella 6.1(a) del dataset MovieRec notiamo immediata-

Tabella 6.2: Dataset MovieRec: Tabelle Metriche di Qualità al variare della soglia per la metrica Gap Posizione Assoluta (a) e Percentuale (b) con l'algoritmo Sarwar

(a) Sarwar-Metrica Gap Posizione Assoluto

	SENZA ANTIRESHUFFLING	GAP ASSOLUTO 15%	GAP ASSOLUTO 20%	GAP ASSOLUTO 25%	GAP ASSOLUTO 30%
RECALL	14,23	13,84	13,81	13,75	13,73
EPR	29,97	29,98	29,98	29,98	29,98
ARHR	5,29	5,74	5,73	5,7	5,67
RESHUFFLING	2,53±0.03	0,96±0.02	0,86±0.02	0,8±0.02	0,75±0.03

(b) Sarwar-Metrica Gap Posizione Percentuale

	SENZA ANTIRESHUFFLING	GAP PERCENTUALE 70%	GAP PERCENTUALE 90%	GAP PERCENTUALE 92,5%	GAP PERCENTUALE 95%
RECALL	14,23	14,05	13,97	13,89	13,86
EPR	29,98	30,01	30	29,98	29,99
ARHR	5,89	5,86	5,85	5,79	5,77
RESHUFFLING	2,53±0.03	1,59±0.02	1,13±0.01	0,94±0.02	0,83±0.02

Tabella 6.3: Dataset Netflix: Tabelle Metriche di Qualità al variare della soglia per la metrica Gap Posizione Assoluta (a) e Percentuale (b) con l'algoritmo Asymmetric SVD

(a) Asymmetric SVD-Metrica Gap Posizione Assoluto

	SENZA ANTIRESHUFFLING	GAP ASSOLUTO 5%	GAP ASSOLUTO 10%	GAP ASSOLUTO 20%
RECALL	5,7	5,66	5,63	5,55
EPR	23,23	23,23	23,23	23,23
ARHR	2,73	2,59	2,56	2,55
RESHUFFLING	3,51±0.04	0,62±0.03	0,39±0.03	0,25±0.03

(b) Asymmetric SVD-Metrica Gap Posizione Percentuale

	SENZA ANTIRESHUFFLING	GAP PERCENTUALE 50%	GAP PERCENTUALE 70%	GAP PERCENTUALE 90%
RECALL	5,7	5,78	5,85	5,87
EPR	23,23	23,25	23,24	23,24
ARHR	2,73	2,74	2,75	2,75
RESHUFFLING	3,51±0.04	2,7±0.04	1,96±0.04	1,06±0.03

mente che i valori migliori delle metriche di qualità si hanno per gli algoritmi di tipo Item-Item Cosine e DR. In particolare per quanto riguarda l Recall il valore più alto (24.58%) si ottiene in corrispondenza dell'algoritmo Item-Item Cosine KNN, il quale presenta anche il miglior ARHR (10.26%). Il valore migliore di EPR si ha invece in corrispondenza dell'algoritmo Item-Item Cosine (10.34%). L'abbassamento di Reshuffling maggiore avviene in corrispondenza degli algoritmi della classe Item-Item. I due Item-Item Cosine in particolare pagano quest'abbassamento con una perdita della Recall di un punto percentuale, valore comunque accettabile. L'algoritmo su cui il metodo correttivo interviene meno (anche se comunque il valore viene dimezzato) è LSA Cosine, dove però notiamo un lieve incremento della Recall. Tra le due metriche di valutazione, quella che appare più efficare, ossia quella che diminuisce maggiormente il valore di Reshuffling è la *Gap Posizione Assoluta*, che però rispetto alla *Gap Posizione Percentuale* ha dei peggioramenti maggiori soprattutto in termini di Recall. Nella Tabella 6.1(b) del dataset Neflix possiamo ancora una volta notare come i valori migliori delle metriche di qualità appartengano all'algoritmo Item-Item Cosine KNN. Gli abbassamenti di Reshuffling sono minori rispetto al dataset precedente ma comunque si possono giudicare positivamente, anche perchè le metriche di qualità non vengono peggiorate significativamente. EPR e ARHR, infatti, si mantengono abbastanza costanti anche applicando il metodo correttivo. La Recall, a parte nell'algoritmo KNN con la prima metrica, si abbassa al massimo di mezzo punto percentuale. Applicando la seconda metrica all'algoritmo Asymmetrica SVD notiamo un lieve aumento della Recall. Nel confronto tra le due metriche possiamo trarre le stesse considerazioni del dataset precedente. Le Tabelle 6.2(a), 6.2(b), 6.3(a) e 6.3(b) mostrano più in dettaglio i risultati ottenuti per ogni singola metrica di valutazione con le diverse soglie utilizzate. Ogni colonna è riferita, a seconda della metrica utilizzata, ad un certo valore di soglia. Vengono riportati i risultati relativi all'algoritmo Sarwar per il dataset MovieRec e all'algoritmo Asymmetric SVD per il dataset Netflix. Possiamo notare, come si poteva prevedere, che all'aumentare della soglia la presenza di Reshuffling diminuisce. Alla stessa manierà, però, le metriche di qualità subiscono lievi peggioramenti di importanza comunque non rilevante. EPR e ARHR peggiorano in termini di centesimi di percentuale, la Recall di decimi di percentuale. Come già detto in precedenza, la metrica *Gap Posizione Percentuale* interviene di meno della *Gap Posizione Assoluta*. Si noti che il valore di Reshuffling ottenuto dalla prima metrica con la soglia più alta è maggiore di quello ottenuto dalla seconda con le soglie minori. Da questo si evince che la seconda metrica è più permissiva rispetto alla prima. Anche in questo caso si può vedere come l'al-

goritmo Asymmetric SVD, nel caso di metrica *Gap Posizione Percentuale*, subisce un lieve aumento di Recall per tutti i valori della soglia.

6.6 Risultati test sui profili ad hoc

Per la parte sul funzionamento di questa metodologia si rimanda alla Sezione 4.3.3. Si riportano le informazioni utili alla lettura delle tabelle. I valori di Reshuffling sono suddivisi nelle seguenti categorie:

- Per Reshuffling minimo **(MIN)** si intende un cambiamento della lista di raccomandazione al più pari al 20% (esempio: in una Top-10 al massimo due item possono cambiare per rimanere nella fascia di Reshuffling minimo).
- Per Reshuffling medio **(MED)** si intende un cambiamento della lista di raccomandazione tra il 30% e il 50%.
- Per Reshuffling alto **(MAX)** si intende un cambiamento della lista di raccomandazione superiore al 50%.

Per ogni cella relativa ad uno specifico algoritmo verrà assegnato un colore: se la cella è grigia significa che le aspettative (in merito al valore di Reshuffling) non sono state rispettate, se è bianca significa che il valore di Reshuffling che ci aspettavamo è stato rispettato o è addirittura migliore. Tra parentesi sono riportate le aspettative e i risultati ottenuti riguardo al contenuto della raccomandazione. Il significato delle sigle contenute nelle tabelle è il seguente:

L Lunghezza del profilo, numero di rating

FC Fattore Comune che raggruppa i film contenuti nel profilo

R Reshuffling

B Risultato ottenuto per l'algoritmo di raccomandazione senza metodo correttivo

M1 Risultato ottenuto utilizzando l'algoritmo correttivo e la metrica *Gap Posizione Assoluta*

M2 Risultato ottenuto utilizzando l'algoritmo correttivo e la metrica *Gap Posizione Percentuale*

A differenza dei profili ad hoc presenti nel Capitolo 4, per ogni caso di test sono presenti tre righe che riportano il valore di Reshuffling in riferimento a quanto segue:

- La prima riga è il risultato dell'algoritmo di raccomandazione.

- La seconda riga è il risultato dell'applicazione dell'algoritmo correttivo con la metrica di *Gap Posizione Assoluta* con un soglia del 20% per il dataset MovieRec e del 10% per il dataset NetFlix.
- La terza riga è il risultato dell'applicazione dell'algoritmo correttivo con la metrica *Gap Posizione Percentuale* con una soglia del 90%.

I valori delle soglie sono stati scelti in base ai risultati ottenuti nella prima metodologia di test, si è cercato di scegliere delle soglie che portassero ad avere degli abbassamenti del valore di Reshuffling senza eccessiva perdita di qualità, misurata attraverso le metriche descritte nei paragrafi precedenti. Con la metrica *Gap Posizione Assoluta* sono state utilizzati due valori di soglia differenti per i dataset, questo perchè NetFlix contiene molti più film di MovieRec. La numerosità di un dataset infatti, per questa metrica è un fattore molto importante. Con la metrica *Gap Posizione Percentuale* la soglia è uguale per entrambi i dataset, questa metrica essendo una percentuale non subisce influenze dalla numerosità di un determinato dataset.

Tabella 6.4: Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
1	-**L**: 5 -**FC** : Genere	MIN (Stesso genere)	Stesso genere(POP)	B	R: 30% (SI)	R: 10% (SI)	R: 10% (NO)	R: 10% (SI)	R: 40% (SI)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (SI)	R: 0% (SI)
				M2	R: 10% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (SI)	R: 0% (SI)
		MIN (Stesso genere)	Altro genere(NOPOP)	B	R: 20% (SI)	R: 20% (SI)	R: 20% (NO)	R: 0% (SI)	R: 0% (SI)
				M1	R: 20% (SI)	R: 20% (SI)	R: 0% (NO)	R: 0% (SI)	R: 0% (SI)
				M2	R: 20% (SI)	R: 20% (SI)	R: 0% (NO)	R: 0% (SI)	R: 0% (SI)
2	-**L**: 3 -**Profilo non popolare** -**FC**: Attore , Genere	MIN (Stesso genere)	Stesso genere e attore (NOPOP)	B	R: 10% (SI)	R: 10% (SI)	R: 10% (SI)	R: 10% (SI)	R: 20% (SI)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
		MIN (Stesso genere)	Stesso genere (NOPOP)	B	R: 0% (SI)	R: 10% (SI)	R: 20% (SI)	R: 10% (SI)	R: 30% (SI)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
		MIN-MED (Stesso genere)	Altro genere, altro attore (NOPOP)	B	R: 10% (SI)	R: 10% (SI)	R: 30% (SI, però presenza di film d'azione)	R: 10% (SI)	R: 10% (SI)
				M1	R: 10% (SI)	R: 10% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
				M2	R: 10% (SI)	R: 10% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)

Tabella 6.5: Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
3	-L: 7 -FC: Genere	MIN (Stesso genere)	Stesso genere (NOPOP)	B	R: 10% (SI)	R: 20% (SI)	R: 0% (SI)	R: 10% (SI)	R: 10% (SI)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 10% (SI)	R: 0% (SI)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 10% (SI)	R: 0% (SI)
		MIN (Stesso genere)	Altro genere (NOPOP)	B	R: 0% (SI)	R: 0% (SI)	R: 10% (SI)	R: 0% (SI)	R: 0% (SI)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)	R: 0% (SI)
4	-L: 4 -FC: Argomento trattato	MIN (Stesso argomento)	Stesso argomento (NOPOP)	B	R: 20% (NO)	R: 50% (NO)	R: 10% (NO)	R: 20% (NO)	R: 20% (NO)
				M1	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 10% (NO)
				M2	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 20% (NO)
		MIN (Stesso argomento)	Altro argomento (NOPOP)	B	R: 10% (NO)	R: 10% (NO)	R: 0% (NO)	R: 10% (NO)	R: 10% (NO)
				M1	R: 10% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
				M2	R: 10% (NO)	R: 10% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
5	-L: 4 **-Profilo non popolare** -FC: Regista	MIN	Stesso regista (NOPOP)	B	R: 40%	R: 50%	R: 40%	R: 70%	R: 30%
				M1	R: 0%	R: 10%	R: 0%	R: 0%	R: 10%
				M2	R: 10%	R: 20%	R: 0%	R: 20%	R: 10%
		MIN	Altro regista (POP)	B	R: 40%	R: 60%	R: 30%	R: 10%	R: 70%
				M1	R: 0%	R: 0%	R: 0%	R: 10%	R: 70%
				M2	R: 0%	R: 0%	R: 0%	R: 10%	R: 70%

Tabella 6.6: Dataset MovieRec: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	Item-Item KNN	Item-Item DR	Sarwar	LSA Cosine
6	-**L**: 3 -**Profilo non popolare** -**FC**: Generi, sequel	MIN (Stessi generi)	Stesso sequel (NOPOP)	B	R: 30% (SI)	R: 30% (SI)	R: 20% (NO)	R: 60% (NO)	R: 30% (NO)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 30% (NO)	R: 30% (NO)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 50% (NO)	R: 30% (NO)
		MIN (Stessi generi)	Stesso generi, no sequel (NOPOP)	B	R: 10% (SI)	R: 10% (SI)	R: 0% (NO)	R: 10% (NO)	R: 30% (NO)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
		MIN-MED (Stessi generi)	Altro genere, no sequel (NOPOP)	B	R: 0% (SI)	R: 0% (SI)	R: 30% (NO)	R: 30% (NO)	R: 0% (NO)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
				M2	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 10% (NO)	R: 0% (NO)
7	-**L**: 5 -**FC**: Popolarità	MIN	(POP)	B	R: 20%	R: 20%	R: 0%	R: 30%	R: 10%
				M1	R: 0%	R: 0%	R: 0%	R: 0%	R: 0%
				M2	R: 10%	R: 0%	R: 0%	R: 0%	R: 0%
		Nessun cambiamento	(NOPOP)	B	R: 0%	R: 0%	R: 0%	R: 0%	R: 100%
				M1	R: 0%	R: 0%	R: 0%	R: 0%	R: 100%
				M2	R: 0%	R: 0%	R: 0%	R: 0%	R: 90%
8	-**L**: 5 -**Profilo non popolare** -**FC**: Non popolarità	Nessun cambiamento	(NOPOP)	B	R: 0%	R: 0%	R: 0%	R: 0%	R: 30%
				M1	R: 0%	R: 0%	R: 0%	R: 0%	R: 0%
				M2	R: 0%	R: 0%	R: 0%	R: 0%	R: 0%
		MED	(POP)	B	R: 80%	R: 60%	R: 80%	R: 80%	R: 10%
				M1	R: 80%	R: 60%	R: 10%	R: 60%	R: 0%
				M2	R: 80%	R: 60%	R: 10%	R: 80%	R: 0%

Test sui profili ad hoc di MovieRec: discussione

I risultati ottenuti per il dataset MovieRec (vedi Tabelle 6.4, 6.5 e6.6) mostrano che nei 24 casi in cui gli algoritmi di raccomandazione non soddisfano le aspettative, l'algortimo correttivo, indistintamente dal tipo di metrica, migliora la raccomandazione in 17 casi. In questi casi molte volte il Reshuffling è azzerato del tutto, mentre alcune volte diminusce fino a rientrare nelle aspettative. Nei restanti 7 casi in cui le aspettative non sono verificate, il Reshuffling rimane invariato, tranne che in 2 casi dove la prima metrica diminuisce leggermente la dinamicità.

Possiamo notare come l'algoritmo che viene migliorato di meno dal metodo di correzione è l'LSA Cosine dove in 3 casi su 6 le aspettative rimangono non verificate. L'Item-Item DR invece, nei 3 casi dove presentava valori eccessivi di Reshuffling, grazie all'algoritmo correttivo restituisce un lista secondo le aspettative.

Per quanto riguarda le tipologie di profilo, quello contenente sequel (profilo 6), dove in 4 casi il valore di Reshuffling era eccessivo, migliora del 50%, mentre il profilo non popolare (profilo 8), una volta che gli si aggiunge un film popolare soltanto in un caso su 4 rientra nelle aspettative. Questo probabilmente per il motivo discusso nel capitolo 3: in un algoritmo Collaborativo, un film popolare può annullare i contributi degli altri film del profilo e quindi portare ad una lista di raccomandazione basata solo su se stesso.

Per quanto riguarda la qualità dei contenuti della raccomandazione, questa non è intaccata praticamente dall'algoritmo correttivo, tranne in un solo caso dove dei film diversi da cioè che si aspettava vengono eliminati dalla lista di raccomandazione.

Nel complesso possiamo giudicare positivamente l'apporto dell'algoritmo correttivo, il quale ha avuto una percentuale di successo del 70%.

Tabella 6.7: Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	KNN	Item-Item DR	Sarwar	Asymmetric SVD
1	-L: 5 -FC: Genere	MIN (Stesso genere)	Stesso genere (POP)	B	R: 60% (SI)	R:40% (SI)	R: 20% (NO)	R: 20% (SI)	R: 50% (NO)
				M1	R: 0% (SI)	R: 10% (NO)	R: 0% (NO)	R: 0% (SI)	R: 0% (NO)
				M2	R: 0% (SI)	R: 10% (NO)	R: 0% (NO)	R: 20% (SI)	R: 30% (NO)
		MIN (Stesso genere)	Altro genere (POP)	B	R: 30% (SI)	R: 70% (SI, però forte presenza di altri generi)	R: 0% (NO)	R: 10% (SI, però forte presenza di altri generi)	R: 50% (NO)
				M1	R: 20% (SI)	R: 20% (NO)	R: 0% (NO)	R: 0% (SI, però forte presenza di altri generi)	R: 0% (NO)
				M2	R: 30% (SI)	R: 70% (SI, però forte presenza di altri generi)	R: 0% (NO)	R: 0% (SI, però forte presenza di altri generi)	R: 10% (NO)
2	-L: 3 -FC: Attore, Genere	MIN (Stesso genere)	Stesso genere e attore (NOPOP)	B	R: 80% (SI)	R: 80% (SI)	R: 60% (SI, però forte presenza di altri generi)	R: 70% (SI)	R: 40% (NO)
				M1	R: 30% (SI)	R: 30% (SI)	R: 0% (SI)	R: 20% (SI)	R: 20% (NO)
				M2	R: 60% (SI)	R: 80% (SI)	R: 30% (SI)	R: 30% (SI)	R: 30% (NO)
		MIN (Stesso genere)	Stesso genere (POP)	B	R: 90% (NO)	R: 90% (NO)	R: 100% (NO)	R: 100% (NO)	R: 50% (NO)
				M1	R: 90% (NO)	R: 60% (NO)	R: 10% (NO)	R: 70% (NO)	R: 0% (NO)
				M2	R: 90% (NO)	R: 90% (NO)	R: 90% (NO)	R: 90% (NO)	R: 20% (NO)
		MIN-MED (Stesso genere)	Altro genere, altro attore (POP)	B	R: 100% (NO)	R: 100% (NO)	R: 90% (NO)	R: 100% (NO)	R: 60% (NO)
				M1	R: 80% (NO)	R: 60% (NO)	R: 0% (NO)	R: 90% (NO)	R: 10% (NO)
				M2	R: 100% (NO)	R: 100% (NO)	R: 60% (NO)	R: 90% (NO)	R: 30% (NO)

Tabella 6.8: Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	KNN	Item-Item DR	Sarwar	Asymmetric SVD
3	-**L**: 7 -**FC**: Genere	MIN (Stesso genere)	Stesso genere (NOPOP)	B	R: 30% (SI, però forte presenza di altri generi)	R: 0% (SI)	R: 10% (NO)	R: 0% (SI)	R: 40% (NO)
				M1	R: 0% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (SI)	R: 0% (NO)
				M2	R: 20% (SI)	R: 0% (SI)	R: 0% (NO)	R: 0% (SI)	R: 10% (NO)
		MIN (Stesso genere)	Altro genere (POP)	B	R: 100% (NO)	R: 100% (NO)	R: 70% (NO)	R: 80% (NO)	R: 70% (NO)
				M1	R: 30% (SI)	R: 60% (NO)	R: 0% (NO)	R: 70% (NO)	R: 0% (NO)
				M2	R: 90% (NO)	R: 100% (NO)	R: 20% (NO)	R: 70% (NO)	R: 30% (NO)
4	-**L**: 4 -**FC**: Argomento trattato	MIN (Stesso argomento)	Stesso argomento (NOPOP)	B	R: 50% (NO)	R: 50% (NO)	R: 10% (NO)	R: 20% (NO)	R: 50% (NO)
				M1	R: 10% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)	R: 0% (NO)
				M2	R: 0% (NO)	R: 10% (NO)	R: 0% (NO)	R: 0% (NO)	R: 20% (NO)
		MIN (Stesso argomento)	Altro argomento (NO POP)	B	R: 100% (NO)	R: 100% (NO)	R: 40% (NO)	R: 50% (NO)	R: 40% (NO)
				M1	R: 10% (NO)	R: 20% (NO)	R: 0% (NO)	R: 30% (NO)	R: 0% (NO)
				M2	R: 70% (NO)	R: 60% (NO)	R: 0% (NO)	R: 20% (NO)	R: 10% (NO)
5	-**L**: 4 -**FC**: Regista	MIN	Stesso regista (POP)	B	R: 60%	R: 50%	R: 20%	R: 80%	R: 50%
				M1	R: 10%	R: 0%	R: 0%	R: 20%	R: 10%
				M2	R: 40%	R: 40%	R: 10%	R: 50%	R: 20%
		MIN	Altro regista (POP)	B	R: 80%	R: 100%	R: 60%	R: 40%	R: 50%
				M1	R: 10%	R: 20%	R: 0%	R: 0%	R: 0%
				M2	R: 60%	R: 100%	R: 40%	R: 10%	R: 0%

Tabella 6.9: Dataset Netflix : risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	KNN	Item-Item DR	Sarwar	Asymmetric SVD
6	-L: 3 -**Profilo Non Popolare** -**FC**: Generi, sequel	MIN (Stessi generi)	Stesso sequel (NOPOP)	B	R: 80%	R: 70%	R: 20%	R: 70%	R: 70%
				M1	R: 20%	R: 0%	R: 0%	R: 0%	R: 10%
				M2	R: 50%	R: 10%	R: 10%	R: 0%	R: 60%
		MIN (Stessi generi)	Stessi generi, no sequel (POP)	B	R: 90% (SI,forte presenza film di uno dei due generi)	R: 90% (SI,forte presenza film di uno dei due generi)	R: 30% (SI,forte presenza film di uno dei due generi)	R: 70% (SI,forte presenza film di uno dei due generi)	R: 40% (SI, film di entrambi i generi)
				M1	R: 20%(SI)	R: 70% (SI)	R: 0%(SI)	R: 10%(SI)	R: 10% (SI)
				M2	R: 70%(SI,forte presenza film di uno dei due generi)	R: 90% (SI,forte presenza film di uno dei due generi)	R: 10%(SI)	R: 50% (SI,forte presenza film di uno dei due generi)	R: 20% (SI)
		MIN-MED (Stessi generi)	Altri generi, no sequel (NOPOP)	B	R: 80% (SI, film di entrambi i generi)	R: 90% (SI, film di entrambi i generi)	R: 30% (SI, film di entrambi i generi)	R: 80% (SI,forte presenza film di uno dei due generi)	R: 50% (SI, film di entrambi i generi)
				M1	R: 30% (SI)	R: 20% (SI)	R: 0%(SI)	R: 20% (SI)	R: 0% (SI)
				M2	R: 50% (SI)	R: 80% (SI)	R: 10%(SI)	R: 50% (SI)	R: 10% (SI)
7	-L: 5 -**FC**: Popolarità	MIN	(POP)	B	R: 50%	R: 50%	R: 20%	R: 50%	R: 30%
				M1	R: 0%	R: 0%	R: 0%	R: 20%	R: 0%
				M2	R: 10%	R: 20%	R: 0%	R: 30%	R: 0%
		Nessun cambiamento	(NOPOP)	B	R: 40%	R: 0%	R: 0%	R: 30%	R: 60%
				M1	R: 0%	R: 0%	R: 0%	R: 20%	R: 10%
				M2	R: 10%	R: 0%	R: 0%	R: 20%	R: 60%

Tabella 6.10: Dataset Netflix: risultati profili ad hoc con algoritmo correttivo (in grigio le aspettative non verificate)

	Caratteristiche Profilo	Aspettative Reshuffling	Azioni		Item-Item Cosine	KNN	Item-Item DR	Sarwar	Asymmetric SVD
8	-L: 5 -**Profilo non popolare** -**FC**: Non popolarità	Nessun cambiamento	(NOPOP)	B	R: 80%	R: 100%	R: 20%	R: 20%	R: 50%
				M1	R: 70%	R: 60%	R: 0%	R: 0%	R: 0%
				M2	R: 70%	R: 80%	R: 0%	R: 10%	R: 10%
		MED	(POP)	B	R: 100%	R: 100%	R: 90%	R: 90%	R: 60%
				M1	R: 100%	R: 50%	R: 0%	R: 50%	R: 0%
				M2	R: 100%	R: 100%	R: 30%	R: 70%	R: 30%

Test sui profili ad hoc di Netflix: discussione

I risultati ottenuti sul dataset Netflix, mostrati nelle Tabelle 6.7, 6.8, 6.9 e 6.10, che in precedenza erano negativi in 74 casi, dopo l'applicazione del metodo correttivo mostrano dei comportamenti differenti a seconda della metrica utilizzata. La metrica *Gap Posizione Assoluta* ottiene risultati positivi in 53 casi su 74, abbassando lievemente il valore di Reshuffling e soltanto in pochi casi abbassandolo del tutto. Tra gli algoritmi che vengono migliorati dal metodo correttivo possiamo individuare ancora l'Item-Item DR. La metrica *Gap Posizione Percentuale* migliora il valore di Reshuffling in 30 casi su 74, abbassando comunque il valore di Reshuffling, ma non abbastanza da soddisfare le aspettative.

Gli algoritmi che presentano i peggiori risultati sono i due della famiglia Item-Item Cosine, dove spesso il valore di Reshuffling rimane troppo eccessivo soprattutto quando si utilizza la seconda metrica.

Analizzando i singoli profili possiamo notare come i due che hanno lunghezza 3 (profili 2 e 6), nonostante la presenza dell'algoritmo correttivo, presentano, valori di Reshuffling elevati, in alcuni casi questo si può spiegare dall'aggiunta di un film popolare, che su un profilo corto può influire molto sulla raccomandazione bypassando il filtraggio effettuato dal metodo correttivo. La qualità dei contenuti della raccomandazione rimane per la maggior parte dei casi identica al caso base, soltanto alcune volte viene leggermente migliorata, senza comunque mai peggiorare.

Nel complesso possiamo giudicare buono l'apporto dell'algoritmo correttivo utilizzando la metrica *Gap Posizione Assoluta* che funziona nel 71% dei casi, mentre scarso l'apporto della seconda metrica che funziona soltanto nel 40% dei casi.

Capitolo 7

Integrazione in un sistema reale

7.1 L'evoluzione temporale

L'applicazione pratica dell'algoritmo correttivo presentato al Capitolo 5 è un problema da non sottovalutare. Se, da un lato, la soluzione più semplice sarebbe quella di applicarlo ad ogni nuova raccomandazione richiesta dall'utente, dall'altro bisogna tener conto dei vari aspetti di un sistema di raccomandazione, che influiscono pesantemente sulle raccomandazioni stesse.

I risultati presentati fin'ora si riferivano a profili reali (ossia presi da dei dataset reali, non inventati) con raccomandazioni generate sulla base di un modello degli item fisso.

Inoltre le tecniche di valutazione precedenti non tengono conto delle caratteristiche dell'evoluzione temporale delle raccomandazioni prodotte.

Quando si applica la tecnica proposta in un sistema reale in evoluzione continua, occore tenere in considerazione che la presenza di Reshuffling tra due raccomandazioni diverse, per un determinato profilo utente, può dipendere da 3 fattori:

- La lunghezza del profilo utente. Questo aspetto è studiato nel Capitolo 4.

- Il modello a cui fa riferimento la raccomandazione. La costruzione del modello su cui si basa la raccomandazione è un'operazione che può richiedere molte risorse a livello computazionale. Si pensi ad esempio ad un sistema con decine di migliaia di utenti e di film. Sicuramente la costruzione di una matrice, come ad esempio l'URM, così grande è un

processo che richiede tempo. E' per questo che la costruzione del modello viene effettuata nella fase di batch come mostrato nella Sezione 2.4. Di conseguenza le raccomandazioni in molti casi devono basarsi su di un modello che non rappresenta la situazione attuale ma che si riferisce ad un istante di tempo passato (più o meno lontano a seconda della frequenza con la quale il sistema di raccomandazione aggiorna il modello degli item). Effettuare una raccomandazione con due modelli diversi può portare ad ottenere diversi risultati. L'algoritmo correttivo potrebbe, quindi, essere anch'esso influenzato dal modello utilizzato.

- I nuovi voti inseriti, nel profilo utente in questione, tra le due raccomandazioni. Il profilo utente è uno dei componenti principali su cui si basa una determinata raccomandazione. L'inserimento di nuovi rating all'interno del profilo può influire pesantemente sulla raccomandazione e di conseguenza anche sulla possibile presenza di Reshuffling. Questo aspetto è stato studiato in parte dalla metodologia di valutazione del precedente capitolo, con un procedimento a ritroso che toglieva i voti dal profilo utente per poi riaggiungerli studiando la presenza di Reshuffling. Non si è tenuto conto però della vera e propria evoluzione temporale del dataset. L'algoritmo correttivo potrebbe, quindi, essere anch'esso dipendente dall'evoluzione temporale dei profili utente.

Nel corso di questo capitolo verranno presentate tre metodologie di test (e i relativi risultati) che valuteranno la presenza di Reshuffling rispetto a: l'effetto dell'evoluzione temporale del modello senza contare l'evoluzione dei profili utenti, l'effetto derivante dall'evoluzione temporale dei profili utente utilizzando un unico modello, l'effetto dell'evoluzione temporale di entrambi gli aspetti. Le misurazioni del Reshuffling avverranno soltanto per i profili considerati *attivi*, dove per attivi si intende l'insieme di utenti che nell'arco della finestra temporale considerata avranno interagito con almeno un item. Nelle simulazione presentate in seguito, si utilizzano 2 finestre temporali, una con lunghezza di settimana (7 giorni) e l'altra di un mese (28 giorni).

Oltre alla presenza di Reshuffling verrà inserita, solo nell'ultimo caso di test, anche una nuova metrica di valutazione. Per poter capire quanto cambiano le raccomandazioni in termini di nuovi film che appaiono nelle liste, si è definita la *Novelty* (lett. Novità) [21]. Nel calcolo della Novelty, anzichè confrontare i film diversi tra due liste di raccomandazione (come facevamo per il Reshuffling), si confronta l'ultima lista di raccomandazione ottenuta, per un determinato utente, con l'intero set di item che sono stati

raccomandati fino al tempo t [21] [6]:

$$\text{Novelty}(L1; N) = \frac{\mid L_1 \setminus A_t \mid}{N} \tag{7.1}$$

Dove N è il numero di film della lista di raccomandazione. A_t è quindi l'unione di tutte le liste di raccomandazione presentate all'utente nel passato. Il concetto di *Novelty* consiste semplicemente nell'insieme di film che non sono mai stati raccomandati ad un utente. Più è alta, più i film raccomandati sono nuovi per l'utente. Bassa, il viceversa.

Questa metrica può essere utile per giudicare la qualità della raccomandazione fornita agli utenti. Per poter valutare attentamente la qualità della raccomandazione modificata dall'algoritmo correttivo verrà utilizzata la metrica di Recall. A differenza dei test effettuati nei capitoli precedenti, la Recall verrà calcolata per mezzo dei film con cui ogni singolo utente ha effettivamente interagito in un momento successivo alla raccomandazione. Si è scelto di utilizzare una finestra temporale di 7 giorni per la valutazione della Recall in accordo con [21].

Nella Sezione 7.2 verrà effettuato un test che studierà l'evoluzione temporale del modello mantenendo statici i profili utente.

Nella Sezione 7.3 verrà effettuato un test che studierà l'evoluzione temporale dei profili utente mantenendo statico il modello. Attraverso questi due test verranno quindi trattati, singolarmente, le ultime due possibili cause della presenza di Reshuffling.

Nella Sezione 7.4, infine, verranno combinati i due precedenti test per studiare il comportamento dell'algoritmo correttivo in un caso reale.

I seguenti test verranno effettuati sul dataset temporale MovieRec descritto nella Sezione 4.2.3. La Figura 7.1 mostra l'evoluzione di questo dataset in termini di film e rating presenti negli istanti di tempo in cui vengono costruite le determinate raccomandazioni, quando viene utilizzata una finestra temporale pari a una settimana. Per le metriche di valutazione verranno utilizzate le seguenti soglie:

- 20% per *Gap Posizione Assoluta*.
- 90% per *Gap Posizione Percentuale*.

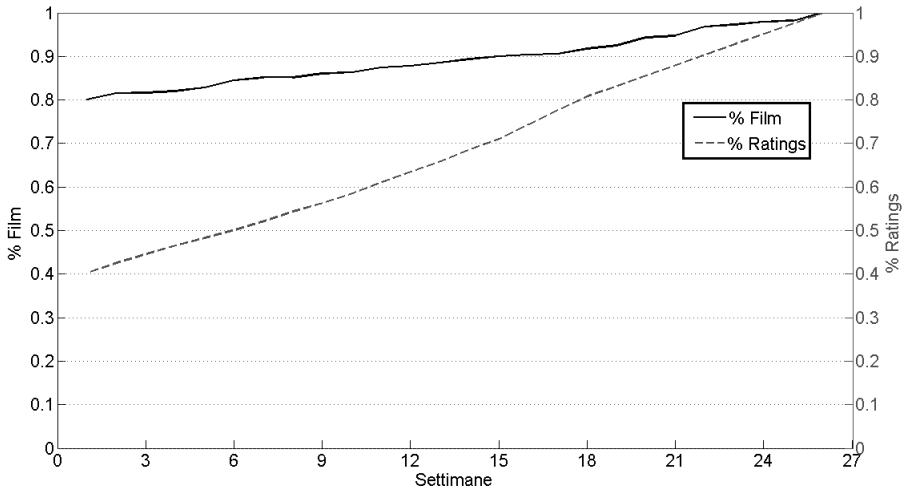


Figura 7.1: Dataset MovieRec Temporale: Evoluzione dei film e dei ratings

7.2 Evoluzione temporale del modello

Questa metodologia di test prevede di mantenere sempre invariati i profili considerati attivi fino all'istante di di inizio delle misurazioni. Viene considerata, quindi, solo l'evoluzione temporale del modello.

Il procedimento del test è il seguente (vedi Figura 7.2): supponendo che la finestra scorrevole abbia come estremo inferiore il tempo t_1 (Tempo 1) e come estremo superiore il tempo t_2 (Tempo 2), con $t_1 < t_2$, il nuovo modello viene costruito ad ogni estremo della finestra scorrevole. In entrambi gli istanti di tempo viene quindi calcolata la raccomandazione con il rispettivo modello per tutti i profili attivi di partenza, tra le due liste ottenute al tempo t_1 e t_2 verrà calcolato il valore di Reshuffling senza e con il metodo correttivo(utilizzato soltanto all'istante t_2), per entrambe le metriche trattate. Tramite i profili attivi nella finestra temporale della Recall (tra t_2 e $t_3 = t_2 + 7$, Tempo 3) verrà valutata la qualità delle raccomandazioni. Una volta effettuati i calcoli, la finestra scorrerà in avanti, procedendo nella stessa maniera, fino all'ultimo istante di tempo utile.

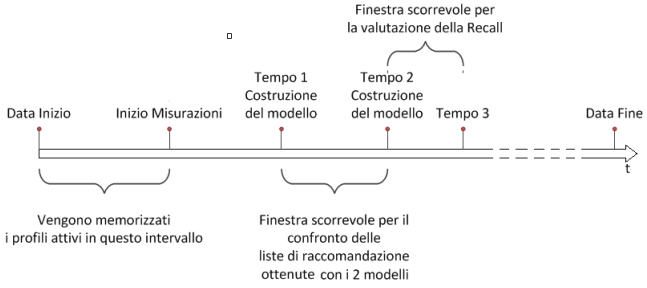


Figura 7.2: Evoluzione temporale del modello con profili fissi

7.2.1 Risultati evoluzione temporale del modello

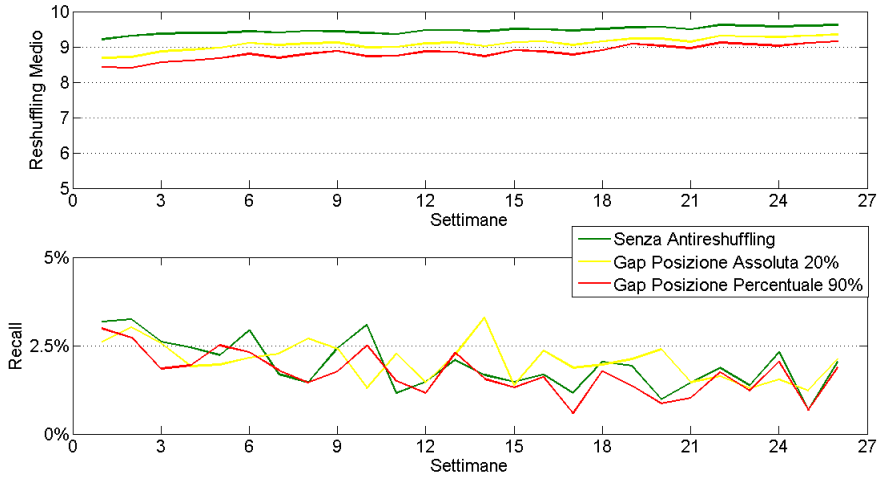
Tabella 7.1: Evoluzione temporale del modello, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

(a) Algoritmo Item-Item Cosine KNN

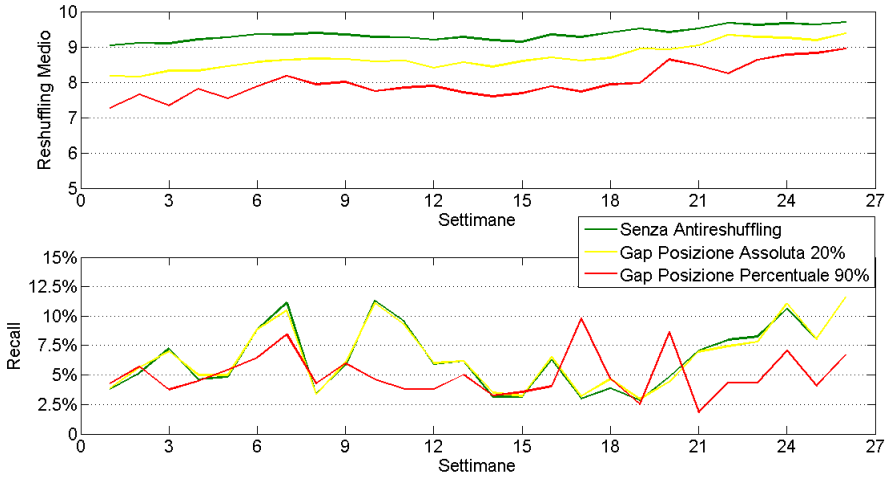
		Reshuffling			Recall
		Media	25esimo	75esimo	
Mese 1	B	8.60	8	10	8.37%
	M1	7.61	6	10	7.91%
	M2	7.12	5	10	6.54%
Mese 2	B	9.05	9	10	3.21%
	M1	8.28	7	10	3.21%
	M2	7.87	7	10	2.91%
Mese 3	B	9.05	9	10	3.02%
	M1	8.45	8	10	3.02%
	M2	8.03	7	10	2.44%
Mese 4	B	9.05	9	10	3.23%
	M1	8.49	8	10	3.53%
	M2	7.97	7	10	2%
Mese 5	B	9.21	9	10	2.60%
	M1	8.67	8	10	2.22%
	M2	8.29	7	10	2.73%
Mese 6	B	9.30	9	10	2.71%
	M1	8.82	9	10	2.71%
	M2	8.44	8	10	1.83%

(b) Algoritmo Sarwar

		Reshuffling			Recall
		Media	25esimo	75esimo	
Mese 1	B	8.62	8	10	7.99%
	M1	7.63	6	10	8.1%
	M2	7.1	5	10	6.19%
Mese 2	B	8.95	8	10	6.2%
	M1	7.82	7	9	6.08%
	M2	7.81	6	10	8.34%
Mese 3	B	8.97	8	10	5.46%
	M1	8.17	7	10	6.91%
	M2	7.45	6	9	6.06%
Mese 4	B	9.11	9	10	15.93%
	M1	8.33	7	10	15.65%
	M2	7.57	6	9	8.6%
Mese 5	B	9.11	9	10	11.87%
	M1	8.5	8	10	12.4%
	M2	7.75	7	9	9.84%
Mese 6	B	9.46	10	10	14.49%
	M1	9.05	9	10	13.92%
	M2	8.34	8	10	9.78%



(a) Algoritmo Item-Item Cosine KNN



(b) Algoritmo Sarwar

Figura 7.3: Evoluzione temporale del modello, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

Commenti sui risultati dell'evoluzione temporale del modello

Nelle Tabelle 7.1(a) e 7.1(b) sono riportati i risultati ottenuti con una finestra scorrevole pari ad un mese (28 giorni). Ogni riga della tabella rappresenta uno dei sei mesi analizzati, ogni sottoriga i risultati ottenuti dall'algoritmo originale e quelli ottenuti dall'applicazione del metodo correttivo con entrambe le metriche. Le due colonne contengono il valor medio di Reshuffling su una Top-10 list con i relativi percentili e la Recall media. Possiamo subito notare come, in questo test, la presenza di Reshuffling sia molto elevata, con valori che si aggirano intorno a otto, nove film su una lista di dieci, per tutti i mesi considerati e per entrambi gli algoritmi analizzati. Quest'elevata dinamicità è causa dell'inserimento continuo di nuovi film nel dataset (come si vede in Figura 7.1). L'evoluzione del modello porta all'integrazione di nuovi film non presenti all'istante iniziale e alla conseguente raccomandazione di essi. Per questo le raccomandazioni, alle iterazioni successive, possono contenere nuovi film che non sono presenti in istanti precedenti causando l'eccessiva dinamicità delle liste.

L'intervento del metodo correttivo porta ad una lieve riduzione del Reshuffling che non si può considerare soddisfacente. Il valor medio infatti viene ridotto di poche unità ma una nota positiva potrebbe essere il fatto che il 25-esimo percentile si abbassa, questo sta ad indicare che il campione (per campione si intende l'insieme dei valori di Reshuffling ottenuti dai vari casi di test) avrà un estremo inferiore minore rispetto al caso originale. Ciò significa che una parte degli elementi del campione ha diminuito il suo valore. Per quanto riguarda la Recall vi sono comportamenti differenti nei due algoritmi utilizzati. Nell'Item-Item Cosine KNN (vedi Tabella 7.1(a)), il valore più alto di Recall si ottiene nel corso del primo mese per poi abbassarsi nei mesi seguenti. L'algoritmo correttivo, che nel primo mese porta ad una perdita di Recall fino a un punto e mezzo percentuale, nei mesi seguenti porta soltanto a lievi diminuzioni. Nell'algoritmo Sarwar (vedi Tabella 7.1(b)) il comportamento della Recall non è uniforme, così come l'effetto del metodo correttivo dove nei diversi mesi la prima o la seconda metrica hanno valori altalenanti. Nel complesso però, qualitativamente, la prima metrica sembra essere la migliore anche se diminuisce il Reshuffling in maniera inferiore rispetto alla seconda. Nelle Figure 7.3(a) e 7.3(b) vengono mostrati i risultati ottenuti con una finestra temporale di una settimana (7 giorni). Per quanto riguarda il Reshuffling le figure confermano quanto detto in precedenza riguardo alle tabelle, ossia che nonostante l'intervento dell'algoritmo correttivo e nonostante il Reshuffling venga ridotto, la sua presenza permane ancora significativamente. C'è da notare come a differenza dei risultati ot-

tenuti nel Capitolo 6 il Reshuffling con la metrica *Gap Posizione Percentuale* sia inferiore rispetto all'altra. Anche in questo caso il comportamento della Recall è molto altalenante. Nel complesso sembra che la metrica *Gap Posizione Assoluta* qualitativamente sia migliore della seconda, anche se interviene meno sulla presenza di Reshuffling.

7.3 Evoluzione temporale dei profili utente

Questa metodologia di test prevede di mantenere sempre un unico modello per effettuare le raccomandazioni. Viene considerata quindi solo l'evoluzione temporale dei profili utente. Il modello viene costruito solo una volta all'istante iniziale delle misurazioni e, ovviamente, deve contenere una determinata quantità di informazioni utili, di conseguenza non verrà costruito alla data d'inizio del dataset a disposizione ma dopo un certo numero di giorni.

Il procedimento di test è il seguente (vedi Figura 7.4): supponiamo la finestra delimitata da un estremo inferiore t_1 (Tempo 1) e un estremo superiore t_2 (Tempo 2), con $t_1 < t_2$. All'interno di questa finestra vengono calcolate le liste di raccomandazione (con algoritmo correttivo solo all'istante t_2) al tempo iniziale e finale per i profili considerati attivi in quest'intervallo per mezzo del modello costruito nel momento iniziale. La valutazione della Recall avviene esattamente come nel caso precedente. Una volta effettuati i calcoli, la finestra scorrerà in avanti, procedendo nella stessa maniera, fino all'ultimo istante di tempo utile.

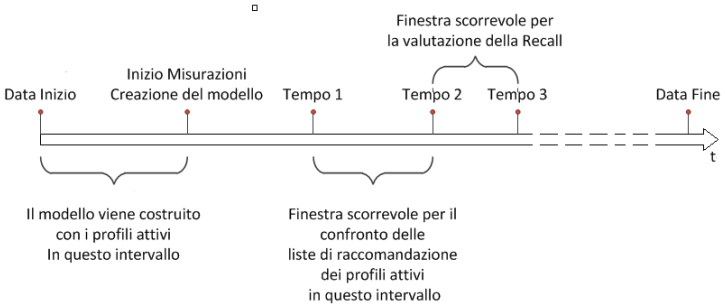


Figura 7.4: Evoluzione temporale dei profili con modello fisso

7.3.1 Risultati evoluzione temporale dei profili utente

Tabella 7.2: Evoluzione temporale dei profili utente, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

(a) Algoritmo Item-Item Cosine KNN

		Reshuffling			Recall
		Media	25esimo	75esimo	
Mese 1	B	2.11	0	3	6.52%
	M1	0.62	1	1	4.58%
	M2	0.82	1	1	4.75%
Mese 2	B	1.96	0	3	3.94%
	M1	0.68	0	0	2.67%
	M2	0.85	0	1	2.82%
Mese 3	B	1.64	0	2	4.43%
	M1	0.64	0	0	3.12%
	M2	0.77	0	1	3.23%
Mese 4	B	1.67	0	3	3.94%
	M1	0.72	0	0	2.83%
	M2	0.85	0	1	2.99%
Mese 5	B	1.57	0	2	4.44%
	M1	0.71	0	0	3.28%
	M2	0.83	0	1	3.53%
Mese 6	B	1.37	0	2	3.99%
	M1	0.62	0	0	2.85%
	M2	0.71	0	0	2.90%

(b) Algoritmo Sarwar

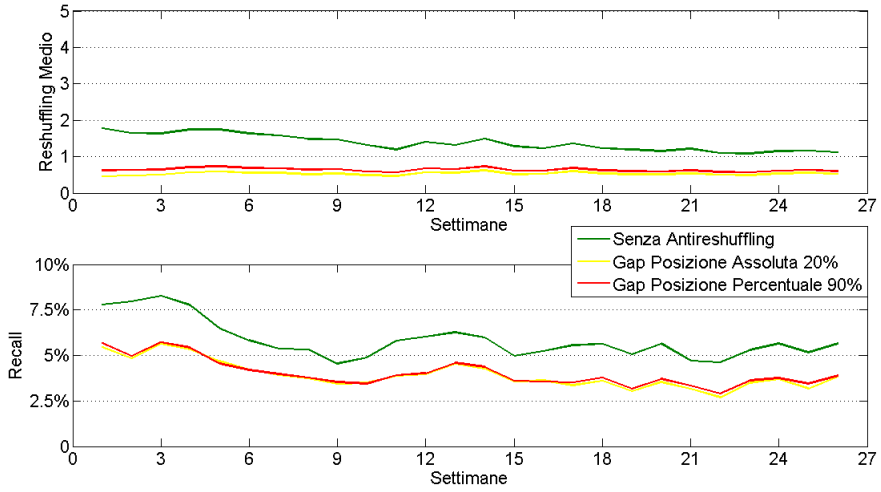
		Reshuffling			Recall
		Media	25esimo	75esimo	
Mese 1	B	2.14	0	3	5.86%
	M1	0.96	0	1	4.32%
	M2	1.1	0	1	4.41%
Mese 2	B	1.92	0	3	4.63%
	M1	0.91	0	1	3.11%
	M2	0.99	0	1	3.16%
Mese 3	B	1.51	0	2	4.76%
	M1	0.72	0	0	3.27%
	M2	0.76	0	1	3.2%
Mese 4	B	1.54	0	2	4.04%
	M1	0.79	0	0	2.81%
	M2	0.79	0	1	2.82%
Mese 5	B	1.4	0	2	4.02%
	M1	0.74	0	0	3.04%
	M2	0.72	0	0	3.05%
Mese 6	B	1.2	0	1	3.63%
	M1	0.63	0	0	2.54%
	M2	0.61	0	0	2.55%

Commenti sui risultati dell'evoluzione temporale dei profili utente

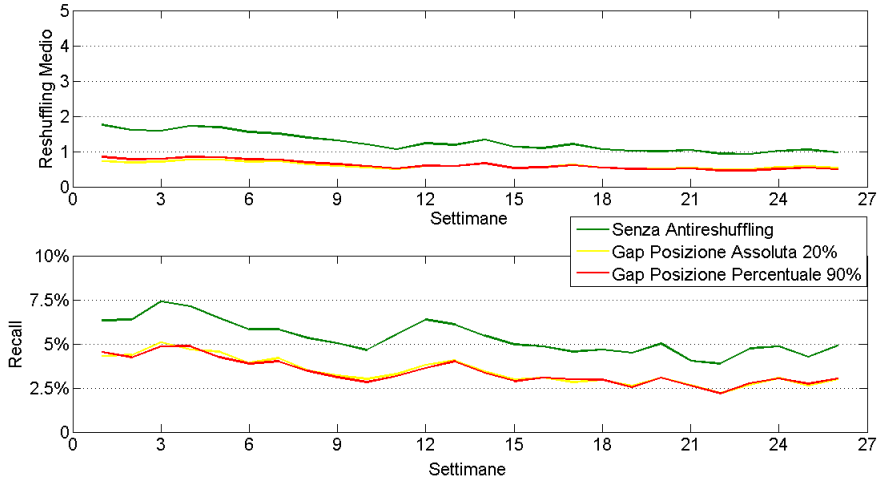
Nelle Tabelle 7.2(a) e 7.2(b) sono ripotati i risultati ottenuti con una finestra scorrevole pari a un mese (28 giorni). Ogni riga della tabella rappresenta uno dei sei mesi analizzati, ogni sottoriga i risultati ottenuti dall'algoritmo originale e quelli ottenuti dall'applicazione del metodo correttivo con entrambe le metriche. Le due colonne contengono il valor medio di Reshuffling su una Top-10 list con i relativi percentili e la Recall media. A differenza del precedente test la presenza di Reshuffling è decisamente minore, mai superiore a due-tre film su dieci.

L'intervento del metodo correttivo porta, in generale, a dimezzare il valore di Reshuffling con un abbassamento del 75-esimo percentile. Questo significa che l'algoritmo correttivo ha diminuito il valore di quegli elementi del campione che originariamente avevano valori più alti. La diminuzione di Reshuffling, in questo caso, è pressochè identica per entrambe le metriche anche se sembra prevalere, di poco, la metrica *Gap Posizione Percentuale*. Per quanto riguarda la Recall, in entrambi gli algoritmi, si riduce nel corso del tempo. L'algoritmo correttivo in tutti i casi porta ad una diminuzione della Recall di un punto, un punto e mezzo percentuale.

Nelle Figure 7.5(a) e 7.5(b) vengono mostrati i risultati ottenuti con una finestra temporale di una settimana (7 giorni). Per quanto riguar-



(a) Algoritmo Item-Item Cosine KNN



(b) Algoritmo Sarwar

Figura 7.5: Evoluzione temporale dei profili utente, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

da il Reshuffling le figure confermano quanto detto in precedenza riguardo alle tabelle. Il comportamento di entrambi gli algoritmi è uniforme. Il Reshuffling viene infatti dimezzato utilizzando entrambe le metriche, con una riduzione di Recall non superiore ai due punti percentuali.

7.4 Evoluzione temporale del sistema di raccomandazione

Questa metodologia di test è la più realistica in quanto viene considerata l'evoluzione sia del modello che dei profili utente. Per questo motivo oltre alla metrica di Recall è stata utilizzata la metrica di Novelty descritta in precedenza. La finestra temporale considerata per il calcolo delle liste di raccomandazione è ancora di lunghezza 7 e 28 giorni, ma per rendere più realistico il test si è ipotizzato che il modello non sia stato costruito nell'esatto istante in cui viene effettuata la raccomandazione. Per questo motivo si è deciso di creare il modello 7 giorni prima di ogni singolo istante in cui vengono calcolate le raccomandazioni.

Il procedimento di test è il seguente (vedi Figura 7.6): supponiamo la finestra delimitata dagli estremi t_1 (Tempo 1) e t_2(Tempo 2) con $t1 < t2$, vengono costruiti due modelli negli istanti $tmod_1 = t_1 - 7$(TempoMod 1) e $tmod_2 = t_2 - 7$(TempoMod 2). Per tutti i profili attivi nell'intervallo considerato vengono calcolate le liste di raccomandazione nei due istanti temporali, con il rispettivo modello, e nell'istante di tempo t_2 viene anche applicato l'algoritmo di correzione. Ovviamente il profilo viene considerato con i rating contenuti in ognuno dei due istanti. Tramite le liste di raccomandazione è possibile quindi calcolare Reshuffling e Novelty. La valutazione della Recall avviene, come nei casi precedenti, nella finestra di lunghezza 7 giorni successiva al tempo t_2. Una volta effettuati i calcoli verrà considerato l'intervallo successivo, procedendo sempre nella stessa maniera, fino all'ultimo istante di tempo utile.

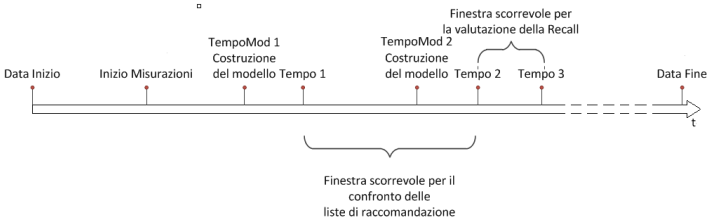


Figura 7.6: Evoluzione temporale del sistema di raccomandazione

7.4.1 Risultati evoluzione temporale del sistema di raccomandazione

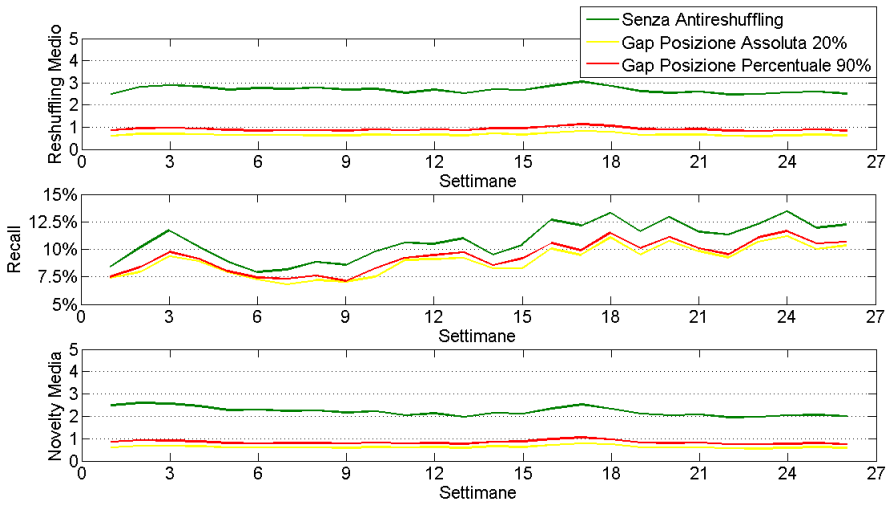
Tabella 7.3: Evoluzione temporale dei sistema di raccomandazione, finestra temporale di una mese. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

(a) Algoritmo Item-Item Cosine KNN

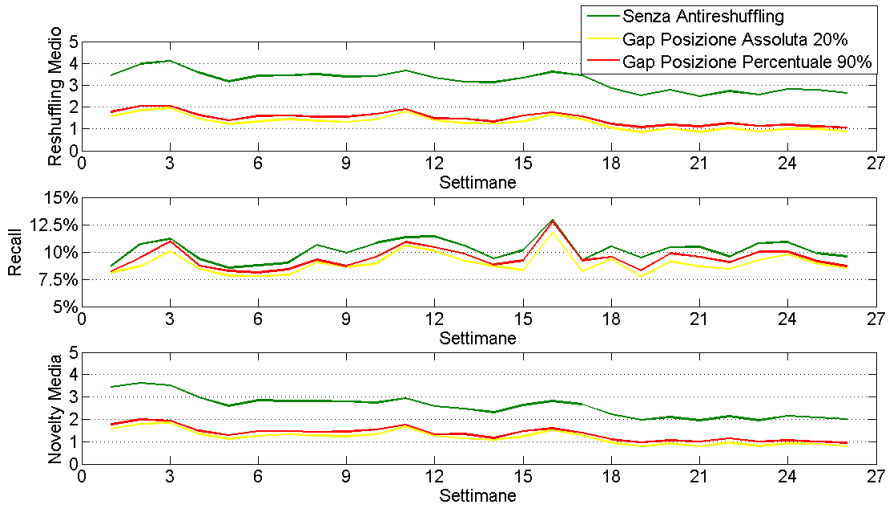
		Reshuffling			Recall	Novelty		
		Media	25esimo	75esimo		Media	25esimo	75esimo
Mese 1	B	3.78	2	5	9.84%	3.78	2	5
	M1	1.53	0	2	8.44%	1.53	0	2
	M2	1.75	0	2	8.50%	1.75	0	2
Mese 2	B	3.62	2	5	7.93%	3.35	2	5
	M1	1.16	0	1	6.05%	1.14	0	1
	M2	1.40	0	2	6.52%	1.36	0	2
Mese 3	B	3.58	2	5	10.16%	3.24	1	5
	M1	1.22	0	1	8.07%	1.20	0	1
	M2	1.50	0	2	8.53%	1.45	0	2
Mese 4	B	3.77	2	5	11.37%	3.41	2	5
	M1	1.49	0	2	9.21%	1.47	0	2
	M2	1.83	0	2	9.85%	1.78	0	2
Mese 5	B	4.04	2	5	12.41%	3.71	2	5
	M1	1.56	0	2	10.16%	1.54	0	2
	M2	2	1	3	10.90%	1.94	1	3
Mese 6	B	3.53	2	5	12.15%	3.15	1	4
	M1	1.20	0	2	9.55%	1.18	0	2
	M2	1.54	0	2	10.35%	1.48	0	2

(b) Algoritmo Sarwar

		Reshuffling			Recall	Novelty		
		Media	25esimo	75esimo		Media	25esimo	75esimo
Mese 1	B	5.42	4	7	8.8%	5.41	4	7
	M1	3.45	2	5	8.31%	3.45	2	5
	M2	3.65	2	5	8.33%	3.65	2	5
Mese 2	B	5.02	3	7	11.43%	4.66	3	6
	M1	2.83	1	4	9.97%	2.78	1	4
	M2	3.05	1	4	10.39%	2.98	1	4
Mese 3	B	4.9	3	6	11.21%	4.42	3	6
	M1	2.61	1	4	9.56%	2.52	1	4
	M2	2.76	1	4	10.16%	2.65	1	4
Mese 4	B	4.83	3	6	11.84%	4.43	3	6
	M1	2.77	1	4	10.93%	2.67	1	4
	M2	2.94	1	4	11.05%	2.82	1	4
Mese 5	B	4.51	3	6	10%	3.99	2	5
	M1	2,42	1	3	9.16%	2.3	1	3
	M2	3.57	1	4	9.77%	2.43	1	4
Mese 6	B	4	2	5	10.53%	3.5	2	5
	M1	2,01	1	3	9.12%	1.93	1	3
	M2	2.24	1	3	9.68%	2.12	1	3



(a) Algoritmo Item-Item Cosine KNN



(b) Algoritmo Sarwar

Figura 7.7: Evoluzione temporale del sistema di raccomandazione, finestra temporale di una settimana. Risultati per gli algoritmi Item-Item Cosine KNN (a) e Sarwar (b)

Commenti sui risultati dell'evoluzione temporale del sistema di raccomandazione

Nelle Tabelle 7.3(a) e 7.3(b) sono ripotati i risultati ottenuti con una finestra scorrevole pari a un mese (28 giorni). Ogni riga della tabella rappresenta uno dei sei mesi analizzati, ogni sottoriga i risultati ottenuti dall'algoritmo originale e quelli ottenuti dall'applicazione del metodo correttivo con entrambe le metriche. Le tre colonne contengono il valor medio di Reshuffling e di Novelty su una Top-10 list con i relativi percentili e la Recall media.

Nella Tabella 7.3(a) sono riportati i risultati ottenuti utilizzando l'algoritmo Item-Item Cosine KNN. Il valore di Reshuffling originale, in tutti i mesi, è pari a tre-quattro film su dieci, quindi una percentuale forse eccessiva. La Recall, dopo un lieve abbassamento tra il primo e il secondo mese aumenta fino alla fine. Il valore di Recall per l'algoritmo KNN viene però minore di quello ottenuto nei test statistici del Capitolo 6. La Novelty nei 6 mesi analizzati è leggermente inferiore al valore di Reshuffling, ciò significa che quando la lista di raccomandazione, cambia, con questo cambiamento, nella maggior parte dei casi, vengono introdotti nuovi film. E' possibile aspettarsi un comportamento del genere, soprattutto in un dataset in evoluzione dove, con il tempo, vengono aggiunti nuovi film in catalogo.

Il metodo Antireshuffling in questo caso porta, con entrambe le metriche, a valori di Reshuffling pari alla metà o ad un terzo del valore originale. E' positivo soprattutto notare l'abbassamento dei percentili, ciò significa una riduzione generale dei valori di molti elementi nel campione. La Recall, in tutti i mesi, utilizzando la metrica *Gap Posizione Assoluta* diminuisce di circa un punto e mezzo, due punti percentuali. Diminuisce anche nel caso si utilizzi la metrica *Gap Posizione Percentuale*, ma, queste diminuzioni, sono sicuramente di minor entità rispetto al caso precedente. La Novelty, come da aspettative, utilizzando l'algoritmo correttivo, si abbassa . Il suo valore diventa di poco inferiore al nuovo valore di Reshuffling, ottenuto applicando il metodo di correzione. Come era immaginabile, l'algoritmo correttivo, per limitare il Reshuffling, elimina dalla lista di raccomandazione i film che non sono ritenuti adatti. Quest'eliminazione coinvolge maggioramente film mai raccomandati prima con una conseguente diminuzione della Novelty.

Nella Tabella 7.3(b) sono riportati i risultati ottenuti utilizzando l'algoritmo Sarwar. Il valore di Reshuffling originale, in tutti i mesi, è elevato: quattro-cinque film su dieci. Si può notare comunque un lieve abbassamento, con l'aumentare dei mesi. La Recall mantiene un comportamento abbastanza costante, con valori più alti nei mesi due e quattro. La Novelty, così come avveniva nell'algoritmo precedente, è leggermente più bassa del Reshuffling,

per cui diminuisce leggermente nel corso del tempo.

Il metodo Antireshuffling in tutti i mesi, porta ad un dimezzamento della dinamicità. Anche se i valori sono ancora abbastanza alti, una diminuzione del 50% è da considerarsi positivo, soprattutto considerando l'abbassamento del 75-esimo percentile, sempre pari almeno a due unità. Ciò significa che buona parte degli elementi del campione, che originariamente avevano un valore eccessivo di Reshuffling, dopo l'applicazione dell'algoritmo correttivo, sono stati abbassati a valori accettabili. La Recall subisce, in tutti i mesi, una lieve ma comunque accettabile diminuzione. A differenza dell'algoritmo precedente, si possono rilevare degli abbassamenti non superiori di un punto percentuale, quindi diminuzioni che si possono considerare positive. La Novelty anche in questo caso diminuisce, a causa del forte intervento del metodo correttivo sui nuovi film che vengono raccomandati.

Nelle Figure 7.7(a) e 7.7(b) sono presenti i risultati ottenuti con una finestra temporale di una settimana. Il primo grafico è riferito al valore medio di Reshuffling ottenuto negli istanti di tempo considerati, il secondo grafico riporta la Recall mentre il terzo la media della Novelty.

Nella Figura 7.7(a) sono riportati i risultati utilizzando l'algoritmo Item-Item Cosine KNN. Si può notare come il Reshuffling rimanga abbastanza costante tra i due-tre film su dieci, lo stesso comportamento, con valori lievemente inferiori, è attribuibile anche alla Novelty. La Recall invece ha un comportamento non uniforme, presentando i valori minimi nelle settimane dalla cinque alla dieci e quelli più alti dalla sedici alla venti e alla settimana ventiquattro.

L'applicazione dell'algoritmo correttivo porta ad una diminuzione del Reshuffling abbastanza costante di circa due unità, con la metrica *Gap Posizione Assoluta* che riduce la dinamicità maggiormente rispetto all'altra metrica. L'influenza del metodo correttivo sulla Novelty avviene nella stessa misura del Reshuffling. Così come avveniva utilizzando una finestra temporale di un mese, sono i film nuovi quelli che l'algoritmo correttivo elimina dalla lista per limitare la dinamicità. La Recall, in questo caso, viene diminuita di circa un punto percentuale con alcuni picchi dove l'abbassamento è anche del 2%.

Nella Figura 7.7(b) sono riportati i risultati utilizzando l'algoritmo Sarwar. Il Reshuffling e la Novelty presentano valori tra i tre-quattro film su dieci. La Recall si mantiene abbastanza costante intorno al 10% con un picco del $12,5\%$ alla settimana sedici.

Il metodo correttivo, anche in questo caso, abbassa Reshuffling e Novelty di circa due unità, portando i valori in un intervallo compreso tra i due-tre film su dieci. Anche in questo algoritmo la diminuzione maggiore di Reshuf-

fling si ottiene con la metrica *Gap Posizione Assoluta*. L'effetto sulla Recall del metodo correttivo, a differenza dell'algoritmo KNN, è inferiore. Con entrambe le metriche, infatti, la Recall viene soltanto leggermente diminuita, e in molti istanti di tempo i suoi valori senza e con metodo correttivo sono praticamente identici.

In conclusione possiamo giudicare positivo l'intervento dell'algoritmo correttivo in merito all'abbassamento del Reshuffling. La diminuzione della Novelty purtroppo è una conseguenza dell'abbassamento delle dinamicità nelle liste di raccomandazione. Dal punto di vista qualitativo, la diminuzione portata alla Recall è leggermente eccessiva utilizzando l'algoritmo Item-Item Cosine KNN, mentre molto positiva con l'algoritmo Sarwar.

Capitolo 8

Conclusioni e sviluppi futuri

L'analisi sulla presenza del Reshuffling ha mostrato la reale esistenza del problema in quasi tutti gli algoritmi testati. In quelli collaborativi soprattutto la dinamicità risultava maggiore quando il film che veniva aggiunto era popolare. In algoritmi di questo tipo, infatti, il Reshuffling è molto legato alla popolarità del profilo e dei film che vengono aggiunti allo stesso.

L'algoritmo correttivo sviluppato ha portato ad un sostanziale abbassamento di Reshuffling come secondo le aspettative. Nei test statistici, dove si sono utilizzate delle metriche di qualità presenti in letteratura, il Reshuffling veniva diminuito senza eccessiva perdita di qualità della raccomandazioni. In alcuni casi la qualità è addirittura aumentata, ma per dimostrare che quest'aumento non è solo una casualità bisognerebbe analizzare singolarmente ogni caso in cui questo accade. Nel complesso i risultati dei test statistici possono essere valutati positivamente, soprattutto per l'algoritmo LSA Cosine, che presentava valori di Reshuffling iniziali abbastanza elevati.

I test personalizzati sui profili ad hoc hanno mostrato come, dopo l'applicazione del metodo correttivo, una maggior percentuale di aspettative fosse soddisfatta. In questo caso non è stato possibile analizzare la raccomandazione tramite metriche di qualità, la valutazione infatti è stata completamente soggettiva.

I test sull'evoluzione temporale hanno dimostrato come l'algoritmo correttivo integrato in un caso reale diminuisca la presenza di Reshuffling, in alcuni casi però con perdite di Recall leggermente eccessive (a seconda dell'algoritmo utilizzato).

I risultati ottenuti, in ogni caso, dipendono sia dalla morfologia del dataset che dall'algoritmo (o dagli algoritmi) di raccomandazione utilizzati. E' compito del provider stabilire quale metrica e quale soglia utilizzare per garantire un certo livello di dinamicità e di qualità della raccomandazione.

Una volta confermata la natura del problema e proposte due metriche correttive, in futuro si potrebbero sviluppare ulteriori metriche di valutazione che considerino, ad esempio l'incremento (o decremento) di punteggio ottenuto da un determinato film. In aggiunta, aumentando la complessità, si potrebbe introdurre una metrica che tenga conto oltre che l'incremento di posizione in classifica, anche la popolarità, sia dei film presenti nella lista di raccomandazione precedente che in quella attuale. Come abbiamo riscontrato in questo lavoro infatti, soprattutto negli algoritmi Collaborativi, film popolari portano ad una maggiore dinamicità, per questo, nel caso in cui si voglia abbassare questa dinamicità, è giusto considerare ogni singolo aspetto. Una metrica di questo tipo potrebbe ad esempio considerare tre fattori:

- la popolarità del film in questione
- la posizione in classifica precedente
- l'incremento di posizione in classifica

Combinando questi tre aspetti si potrebbe creare una metrica flessibile e che tenga conto di diversi fattori che influiscono significativamente sulla raccomandazione (e quindi non si farebbe più riferimento solo al singolo incremento di posizione in classifica o all' incremento di punteggio). Dando poi diverso peso ad ognuno di essi si potrebbe in qualche modo sbilanciare la raccomandazione favorendo ad esempio i film popolari oppure i film con un incremento di posizione elevato etc.

Bibliografia

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender Systems Handbook*. 2010.

[3] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*, pages 43–52. IEEE Computer Society, 2007.

[4] Elisa Campochiaro, Paolo Cremonesi, and Roberto Turrin. Analysis of recommender systems based on implicit datasets. Technical report, Politecnico di Milano, 2008.

[5] Kian Ming Adam Chai, Hai Leong Chieu, and Hwee Tou Ng. Bayesian online classifiers for text classification and filtering. In *SIGIR*, pages 97–104. ACM, 2002.

[6] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian Mackinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666, New York, NY, USA, 2008. ACM.

[7] Paolo Cremonesi and Roberto Turrin. Analysis of cold-start recommendations in iptv systems. In Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme, editors, *RecSys*, pages 233–236. ACM, 2009.

[8] Paolo Cremonesi and Roberto Turrin. Time-evolution of iptv recommender systems. In *EURO ITV 2010, Tampere, Finland*. ACM, June 2010.

[9] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. L, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems. Springer-Verlag*, 22/1, 2004.

[11] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1996.

[12] Miha Grcar, Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. knn versus svm in the collaborative filtering framework. *Data Science and Classification*, pages 251–260, 2006.

[13] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 263–272, 2008.

[15] Parry Husbands, Horst Simon, and Chris Ding. On the use of singular value decomposition for text retrieval. In *Proc. of SIAM Comp. Info. Retrieval Workshop*, 2001.

[16] Jens F. Jensen. Interactive television - a brief media history. In Manfred Tscheligi, Marianna Obrist, and Artur Lugmayr, editors, *EuroITV*, volume 5066 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2008.

[17] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995*, pages 1137–1145, 1995.

[18] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. Technical report, AT&T Labs Research, 180 Park Ave, Florham Park, NJ 07932, 2008.

[19] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. AT&T Labs , Research 180 Park Ave, Florham Park, NJ 07932, August 2008.

[20] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[21] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems.

[22] Manos Papagelis and Dimitris Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. In Matthias Klusch, Sascha Ossowski, Vipul Kashyap, and Rainer Unland, editors, *CIA*, volume 3191 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2004.

[23] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, chapter 10, pages 325–341. 2007.

[24] Gerard Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer.* Addison-Wesley, 1988.

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[26] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM, Addison-Wesley.

[27] E. Vozalis and K. G. Margaritis. Analysis of recommender systems' algorithms. In *The 6th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA), Athens, Greece*, 2003.

[28] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM Press.