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

Spiegazioni e confidenza nei sistemi di raccomandazione

VP-Lab
Laboratorio di Valutazione delle Prestazioni

Relatore: Prof. Paolo CREMONESI
Correlatore: Prof. Roberto TURRIN

Tesi di Laurea di:
Luca Chiodi, matricola 720916

Anno Accademico 2009-2010

A chi mi è stato vicino in tutti questi anni...

Sommario

I sistemi di raccomandazione permettono di selezionare, all'interno di un ampio catalogo, un numero limitato di prodotti (ad esempio film, libri, ..., detti generalmente *item*) personalizzati sulla base delle preferenze dell'utente attivo. La ricerca in questo ambito si è sempre concentrata sulla qualità delle raccomandazioni di questi sistemi, tralasciando un aspetto fondamentale: la fiducia che un utente deve avere verso questi ultimi. Riuscire a convincere un utente a fidarsi delle raccomandazioni che gli sono state fatte avrebbe un duplice beneficio: da una parte invoglierebbe l'utente ad acquistare di più, dall'altra aumenterebbe la sua soddisfazione. In questo lavoro vengono individuate due possibili informazioni da presentare all'utente per renderlo più consapevole sulle raccomandazioni proposte: le spiegazioni e la confidenza.

La prima cerca di spiegare agli utenti le raccomandazioni che gli sono state fatte legando gli *item* proposti agli *item* precedentemente visti dall'utente. L'altra informazione, la confidenza, è intesa come grado di certezza che questi *item* possano piacere ad un utente e può essere utilizzata come calcolo della bontà delle raccomandazioni.

Ringraziamenti

Ringrazio innanzitutto Roberto che mi ha seguito e aiutato per quasi un anno e mezzo e il professor Cremonesi che mi ha dato l'opportunità di lavorare in un ambito estremamente attuale e d'avanguardia, quale l'IPTV. Ringrazio le persone che sono passate in tutto questo tempo dal VP-lab, senza le quali le lunghe giornate passate davanti a Matlab sarebbero state sicuramente più lunghe. Ringrazio inoltre i miei genitori che mi hanno dato la possibilità di portare a termine gli studi.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni si è assistito al diffondersi di nuove tecnologie, quali VoD (Video On Demand) e IPTV (IP Television), che consentono ad un utente di poter accedere ai programmi che desidera in qualsiasi momento, direttamente dalla televisione di casa. Grazie alle capacità interattive di tali tecnologie, è possibile collezionare informazioni relative all'uso del sistema da parte degli utenti (ad esempio, quali film hanno guardato). Queste informazioni consentono ai provider VoD e IPTV di analizzare i gusti dei propri clienti in modo da fornire loro contenuti che li soddisfino il più possibile e quindi indurli a fare maggior uso del sistema. La tecnologia che consente di analizzare i gusti degli utenti e proporre contenuti personalizzati è fornita dai sistemi di raccomandazione. Spesso l'unico input di tali sistemi è il profilo dell'utente, ovvero un insieme di valutazioni (note come *rating*) raccolte implicitamente od esplicitamente, relative ad *item* che l'utente ha considerato, siano essi libri, film, cd musicali, ecc. Nello specifico, i sistemi di raccomandazione usati in questo lavoro sono in grado di suggerire agli utenti dei film che potrebbero essere di loro interesse e il profilo utente è composto da una lista di film che l'utente stesso ha guardato e dai relativi *rating*.

Esistono due principali tipologie di sistemi di raccomandazioni:

Content-Based: sistemi che suggeriscono film con un contenuto simile al contenuto dei film già visti dall'utente (ad esempio stessi attori o stesso genere)

Collaborativi: sistemi che suggeriscono film in base alle preferenze degli utenti simili all'utente corrente.

I sistemi di raccomandazione devono essere in grado di dare consigli agli

utenti in *real-time*: per fare ciò non si basano su tutte le informazioni in loro possesso riguardanti tutti gli utenti (in quanto consistono in una mole di dati troppo elevata per essere analizzata in tempo reale), ma si basano su un modello, ossia una rappresentazione compatta dei dati, e sul profilo degli utenti, ossia sugli *item* con cui gli utenti hanno interagito in qualche modo.

Una volta eseguita la raccomandazione, si ha una lista di film da suggerire all'utente, la cosiddetta **TopN**, con N generalmente compreso tra cinque e dieci.

Il problema che si è posto fin da subito nell'utilizzo di questi sistemi consiste nel fatto che l'utente, una volta ricevuta la lista di raccomandazione, non ha modo di sapere come l'algoritmo impiegato abbia stilato quella lista TopN, ossia viene chiesto all'utente di fidarsi di un sistema che funziona come una *black-box*. Se si riuscisse quindi a spiegare ad un utente perché gli sia stato suggerito di guardare un determinato film, ciò avrebbe un duplice beneficio: convincere l'utente ad utilizzare il sistema di raccomandazione (beneficio in termini di *promotion*) e permettergli di prendere decisioni più accurate sul film da vedere (beneficio in termini di *satisfaction*).

In modo simile, è utile per l'utente sapere la confidenza che l'algoritmo ha nella raccomandazione, ovvero quanto l'algoritmo (sulla base delle informazioni a sua disposizione) è sicuro dell'*item* proposto: per l'utente rappresenta infatti un altro elemento per decidere se guardare o meno un certo film. Oltre ad essere un'informazione utile per l'utente, la confidenza può essere utilizzata dal sistema stesso per decidere in modo più accurato quali *item* consigliare. Ad esempio, a parità di *rating* stimato (cioè quanto l'algoritmo pensa che un *item* sarà di gradimento per l'utente), si potranno preferire gli *item* con una confidenza più elevata.

Entrambe le esigenze (spiegazione e confidenza degli *item* raccomandati) sono state affrontate in questo lavoro tramite un approccio innovativo basato su modifiche forzate del profilo utente, ossia aggiungendo o rimuovendo da questo profilo dei *rating* relativi ad alcuni film: in particolare sono stati rimossi dei *rating* per cercare di dare una spiegazione ai film consigliati, mentre ne sono stati aggiunti per dare un valore di confidenza alle raccomandazioni.

1.1 Struttura della tesi

La tesi è strutturata nel seguente modo:

nel capitolo 2 vengono presentati gli algoritmi di raccomandazione esistenti e i diversi tipi di dataset su cui operano. Viene inoltre mostrato lo stato dell'arte per quanto riguarda le spiegazioni e la confidenza delle raccomandazioni.

Nel capitolo 3 vengono mostrati i dataset e gli algoritmi usati in questa tesi e viene data una spiegazione del loro funzionamento.

Nel capitolo 4 viene mostrato il problema della perturbazione del profilo utente.

Nel capitolo 5 vengono introdotte le due metodologie usate per cercare di spiegare i film raccomandati, con i relativi risultati ottenuti.

Nel capitolo 6 vengono definiti i due metodi usati per calcolare la confidenza delle raccomandazioni e vengono mostrati i relativi risultati.

Nel capitolo 7 vengono introdotte e le metriche di *Recall* e *Fall-out* per valutare i risultati ottenuti riguardanti la confidenza.

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

Capitolo 2

Stato dell'arte

In questo capitolo verranno mostrati gli elementi di un sistema di raccomandazione, la loro architettura e le informazioni che usano (*dataset*) per consigliare *item* agli utenti. Verranno inoltre mostrate le due famiglie di algoritmi, il loro funzionamento e i relativi limiti, e infine verranno mostrati i risultati delle ricerche effettuate sullo stato dell'arte del problema delle spiegazioni e della confidenza, dai quali abbiamo preso spunto per il nostro lavoro.

2.1 Introduzione

Negli ultimi anni è stata sviluppata una nuova tecnologia per vedere la televisione, ossia i sistemi di *Video On Demand:* questi sistemi offrono all'utente la possibilità di guardare contenuti di suo interesse in formato digitale, non necessariamente in diretta ma quando ne ha voglia, essendo sempre disponibili in enormi basi di dati, e il suo sviluppo è stato possibile grazie al concetto di IPTV. Questo metodo interattivo di guardare la TV sfrutta come canale di comunicazione la rete globale, ossia internet: questo può avvenire sia collegandosi dal proprio PC al sito internet del provider di servizi VoD, oppure tramite un apposito decoder da collegare al televisore, chiamato *Set-top Box*, che si occuperà della ricezione dei contenuti digitali e della loro conversione in modo da poter essere visti su un televisore.

Esistono tre modi diversi di vedere i contenuti digitali presenti sulla rete:

1. **Streaming:** il video verrà scaricato mentre viene visto dall'utente.

2. **Download e streaming:** il video viene visto solo dopo che ne è stata scaricata una parte.

3. **Download:** il contenuto viene scaricato e l'utente potrà guardarlo quando preferisce.

La nascita dell'IPTV [12] è stata resa possibile grazie al continuo sviluppo di tecnologie che permettono di scambiarsi una quantità sempre maggiore di dati tramite internet, ossia la banda larga: è stato inoltre reso possible aumentare la quantità di informazioni trasmesse tramite i vecchi canali analogici, introducendo la possibilità da parte dell'utente di comunicare con il sistema, ossia di mandare dati (quali ricerche) invece che riceverli solamente. Queste ricerche vengono effettuate sfruttando due tecnologie:

- ***Information Retrieval* (IR):** consiste nel recupero mirato di informazioni in formato elettronico. Viene posta al sistema una *query* contenente la parola chiave da ricercare e il sistema restituisce un oggetto che contiene quella parola. Questa ricerca, nell'ambito dei sistemi di VoD, ha lo scopo di trovare un contenuto per un utente in base ai suoi interessi (espressi dalla parola chiave): in quanto l'unico mezzo a disposizione dell'utente per interagire con il sistema è il telecomando, questa ricerca risulta molto scomoda e complicata, data l'enorme quantità di risultati che è possibile ottenere.

- ***Information Filtering* (IF):** il filtraggio delle informazioni può essere considerato come un'evoluzione dei sistemi di IR: a differenza di quest'ultimo infatti considera nella ricerca anche il soggetto che l'ha effettuata, facendo in modo di ridurre i risultati disponibili cercando di lasciare solo quelli che possono in qualche modo interessare all'utente. Questa tecnica è alla base del funzionamento dei sistemi di raccomandazione.

Questi sistemi di raccomandazione stanno diventando sempre più importante negli ambiti di *e-commerce* in quanto aiutano l'utente a destreggiarsi tra le migliaia di offerte disponibili e a scegliere quelle di suo interesse, facendogli impiegare meno tempo per la ricerca e cercando di renderlo più soddisfatto del risultato della ricerca stessa. Dal lato del provider dei servizi invece questi sistemi hanno il beneficio di spingere l'utente a fare più acquisti e quindi a spendere di più. Gli oggetti su cui operano questi sistemi

sono i profili degli utenti: questi profili sono una collezione di informazione riguardanti articoli (comunemente chiamati *item*) descritti da un insieme di informazioni (metadati) con i quali l'utente ha avuto una qualche sorta di interazione: nei sistemi di raccomandazione da noi analizzati questi *item* sono i film e le informazioni su di essi che si hanno sono il fatto che l'utente abbia visto oppure no un certo film ed eventualmente il voto che gli ha attribuito, mentre i metadati sono ad esempio parole del titolo del film, il cast, parole della trama, ...

Le raccomandazioni che vengono calcolate per un utente si basano soltanto su questi dati a disposizione del sistema, in quanto un utente generalmente quando inizia ad usare un sistema di raccomandazione sarà poco propenso a lasciare informazioni personali quali interessi, gusti e dati anagrafici.

2.2 Dataset

Un dataset rappresenta la totalità delle informazioni in possesso di un sistema di raccomandazione. E' composto da due matrici:

- Matrice URM (*User Rating Matrix*): è una matrice che racchiude informazioni sui film visti da tutti gli utenti del sistema: ipotizzando che nel sistema siano presenti m utenti e n *item*, la matrice URM ha dimensione $m \times n$: ogni riga rappresenta quindi il profilo utente (ossia le informazioni sui film visti da quell'utente) ed ogni colonna rappresenterà un film presente nel sistema. Questa matrice è l'unico input per un algoritmo di tipo collaborativo: contiene infatti tutte le informazioni necessarie per fare una raccomandazione. Un esempio di matrice URM è mostrato in figura 2.1, dove l'elemento $r_{i,j}$ rappresenta la preferenza che l'utente i ha verso l'*item* j (questa preferenza può essere esplicita o implicita, come verrà mostrato nelle sezioni 2.2.1 e 2.2.2).

- Matrice ICM (*Item Content Matrix*): è la matrice che contiene i metadati relativi ad ogni *item* presente in catalogo [4]. I metadati derivano dall'analisi degli *item* e nel caso dei film possono essere attore, una parola del titolo, il protagonista, il genere. La matrice ICM potrebbe non essere accurata in quanto ogni giorno vengono inseriti nel catalogo nuovi film che possono comprendere nuovi metadati: va quindi aggiornata di recente per far sì che la qualità delle raccomandazioni non cali,

dato che questa matrice è usata solo in sistemi di raccomandazione di tipo *Content-Based*. Un'esempio di ICM è mostrato in figura 2.2: l'elemento $P_{s,j}$ rappresenta il peso che il metadato s ha nel film j.

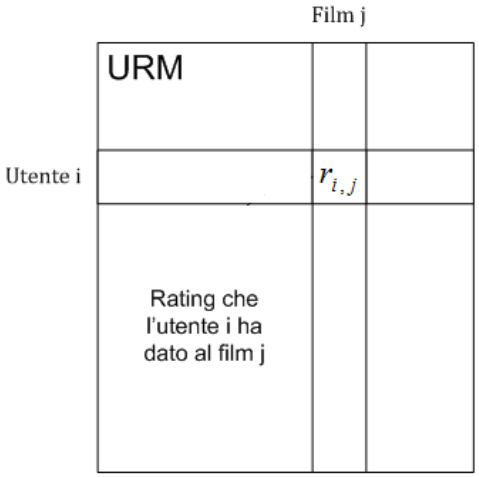


Figura 2.1: Esempio di matrice URM

I voti presenti nella matrice URM possono essere di due tipi: espliciti o impliciti.

2.2.1 Dataset espliciti

Un dataset si dice esplicito se all'interno della matrice URM è possibile trovare valori (solitamente compresi tra 1 e 5, ma anche tra -2 e +2) che rappresentano il voto che un utente ha dato a quell'*item*. Sebbene una raccomandazione basata su *rating* dati esplicitamente dagli utenti possa sembrare più accurata di una effettuata su *rating* impliciti, ci sono alcuni problemi da considerare: in primo luogo la differenza del criterio di giudizio tra i diversi utenti, ad esempio per un utente una votazione pari a 3 su 5 può voler esprimere che il film gli è piaciuto, mentre per un altro potrebbe voler dire che il film non è stato di suo gradimento [4]. Inoltre certi algoritmi di raccomandazione cercano di predire il voto che un utente potrebbe dare ai film in catalogo: se questo voto è dato su una scala differente da quella usata in fase di raccolta delle votazioni dei film visti, potrebbe essere necessario del lavoro aggiuntivo per tradurre questi valori su una scala comune.

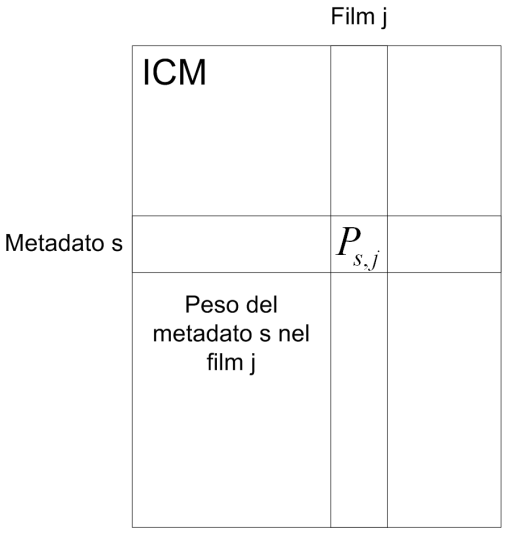


Figura 2.2: Esempio di matrice ICM

2.2.2 Dataset impliciti

I dataset impliciti cercano di raccogliere informazioni sui gusti di un utente in base al suo comportamento: se un utente compra un film tramite il servizio di *Video On Demand*, il sistema segnerà nella matrice URM che c'è stata una sorta di interazione tra quell'utente e quel film, solitamente con un valore pari ad 1; se invece non c'è stato nessun tipo di interazione sarà presente uno 0. Il problema di questa tipologia di *rating* è rappresentato dal fatto che può succedere che un utente compri un film e che dopo averlo visto scopra che non era affatto di suo interesse: in questo caso il sistema, limitandosi solo a registrare l'acquisto del film, crederà che a quell'utente sia piaciuto il film. Solitamente si usano sistemi con *rating* impliciti quando non si è in grado di chiedere agli utenti il livello di gradimento di un film (o comunque di un qualsiasi *item*). Da notare che ogni *rating* esplicito costituisce anche un'informazione implicita.

2.3 Architettura di un sistema di raccomandazione

Un sistema di raccomandazione deve essere in grado di fornire consigli immediati all'utente, appena quest'ultimo li richiede; data la quantità di dati di

cui si compongono i dataset di questi sistemi, è impossibile pensare di ricavare le raccomandazioni direttamente dalle matrici URM e ICM e di mostrare i risultati in pochissimi istanti all'utente: per questo motivo, un sistema di raccomandazione è composto da due fasi, una *Batch* e una *Real-time* (figura 2.3).

Durante la fase batch, il sistema crea un modello a partire dalla matrice URM (e/o ICM, a seconda dell'algoritmo utilizzato) che altro non è che una sorta di visione compatta dei dati a disposizione. Questo modello solitamente si calcola nei momenti in cui il sistema è poco utilizzato, in quanto la sua creazione comporta un intenso carico di lavoro e non è istantanea, dovendo analizzare centinaia di migliaia di utenti e diverse migliaia di film; il modello è necessario anche in quanto un utente solitamente viene in contatto con una bassissima percentuale di film presenti nel catalogo, quindi le matrici dei dataset sono caratterizzate da un'altissima sparsità (oltre il 99% nella maggior parte dei casi reali), è quindi inutile utilizzarle per intero ad ogni raccomandazione. La matrice URM è in continua evoluzione, basta pensare alla frequenza con cui vengono aggiunti al catalogo nuovi film e nuovi utenti decidono di utilizzare il sistema: per questo motivo il calcolo del modello va effettuato frequentemente, altrimenti rischia di essere una rappresentazione di una matrice non aggiornata, il che comporterebbe una cattiva qualità delle raccomandazioni.

Nella fase di *Real-time* invece viene calcolata la lista di raccomandazione TopN da mostrare all'utente, grazie alle informazioni relative all'utente stesso (il suo profilo utente) e al modello precedentemente creato. Questo calcolo deve impiegare il minor tempo possibile per far sì che l'utente non perda fiducia nel sistema di raccomandazione [4].

2.4 Algoritmi di raccomandazione

In questa sezione verrà illustrato il funzionamento delle due famiglie di algoritmi di raccomandazione, ossia quelli *Content-Based* e quelli Collaborativi

2.4.1 Content-Based

Gli algoritmi *Content-Based* generano delle liste di raccomandazione basandosi sui contenuti stessi dei film da consigliare [17]: viene infatti creato un profilo utente che corrisponde ai gusti dell'utente, dedotti dai film che ha già

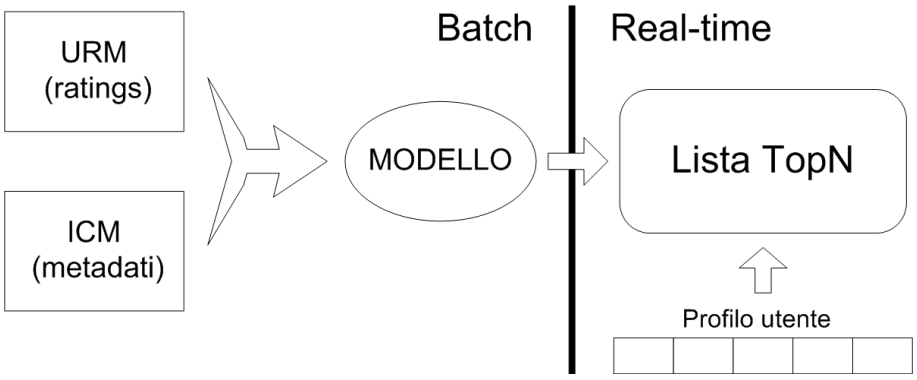


Figura 2.3: Architettura di un sistema di raccomandazione

visto, ed in base a questi gusti si cercano nel catalogo a disposizione film con contenuti simili. Questo tipo di algoritmo necessita di una fase preliminare alla creazione del modello: la creazione della matrice ICM. La matrice ICM è una matrice che contiene, per ogni film presente nel dataset, un elenco di pesi relativi ai vari metadati: questi metadati possono essere nomi di attori, registi, comparse, oppure parole della trama, del titolo, il genere del film e altre ancora. Questo tipo di algoritmo, proprio perché si basa sulle preferenze di un utente per effettuare la raccomandazione, potrebbe a prima vista sembrare molto funzionale, in realtà ha alcuni limiti [2]:

- La qualità e la completezza delle fonti da cui si ricavano le informazioni sui film si riflettono molto sulla qualità della matrice ICM: infatti per certi film (poco conosciuti) si avranno a disposizione delle informazioni in quantità ridotta rispetto a quelle ricavabili per film molto più famosi.

- La qualità della matrice ICM a sua volta si riflette sulla qualità delle raccomandazioni: infatti dopo aver raccolto le informazioni sui film, vanno adeguatamente filtrate per eliminare eventuali dati non necessari.

- Ad un nuovo utente non si potrà consigliare nessun film, in quanto non si è a conoscenza dei suoi gusti (problema della *Cold-start* [6]).

- Ad un utente verranno sempre consigliati film relativi al genere di suo interesse, e mai film di un altro genere, che potrebbero comunque interessargli.

- Se si trovano due o più film con caratteristiche molto simili, per l'algoritmo risultano identici, quindi non c'è distinzione tra un film che potrebbe piacere ad un utente ed uno che potrebbe non piacergli, se presentano ad esempio lo stesso genere e lo stesso cast.

2.4.2 Collaborativi

Gli algoritmi collaborativi simulano in un certo senso un passaparola tra amici, ad esempio quando un utente non sa quale film noleggiare chiede consiglio ad amici con gusti simili o cerca su Internet qualche recensione [19]. Partendo dai film visti da un utente, questa tipologia di algoritmo inserisce quest'ultimo in un cluster di utenti con gusti simili, quindi gli consigli un film che ad altri utenti (i cosiddetti vicini) è piaciuto [8]. Gli algoritmi collaborativi si basano quindi su due concetti [4]:

- Concetto di vicinanza tra utenti: utenti con gusti simili tendono a votare i film in una maniera simile.
- Concetto di vicinanza tra *item*: i film correlati tra loro vengono votati dagli utenti in modo simile.

A differenza degli algoritmi *Content-Based*, quelli collaborativi sono in grado di consigliare ad un utente un film diverso da tutti quelli presenti nel suo profilo, creando una sorta di effetto a sorpresa [10]: con gli algoritmi *Content-Based* invece l'utente sarà quasi sempre in grado di capire il perché di una raccomandazione, notando somiglianze tra i contenuti di suo interesse e quelli dei film consigliati, e quindi sarà meno propenso a stupirsi davanti una raccomandazione. Anche questi tipi di algoritmi presentano però delle limitazioni [2]:

- Per un nuovo utente non sarà possibile effettuare nessuna raccomandazione in quanto non avendo visto nessun film, non può essere inserito in nessun cluster di utenti.
- Un nuovo film non potrà mai essere consigliato, in quanto non avendo ricevuto voti non sarà confrontabile con nessun altro film presente nel catalogo.
- La qualità della raccomandazione dipende molto dalla quantità di film votati da un utente: se infatti un utente ha votato pochi film, sarà difficile identificare un cluster adeguato per lui, in quanto i suoi interessi

potrebbero corrispondere (da quel poco che si sa) con quelli di utenti presenti in più cluster.

- E' molto difficile calcolare raccomandazioni per utenti con gusti particolari sempre per lo stesso motivo: risulta difficile associarli ad un gruppo di utenti.

All'interno di questa categoria di algoritmi è possibile distinguere sue sottocategorie [27]:

- ***User-Based:*** il concetto fondamentale è l'utente, si cercano di creare relazioni di similitudine tra utenti: se in un sistema sono presenti m utenti, il modello creato avrà dimensione $m \times m$, in cui sia le righe che le colonne rappresentano gli utenti. Ogni utente della URM viene quindi rappresentato da un vettore in uno spazio ad n dimensioni, con n numero di film presenti nella matrice URM: saranno considerati simili quegli utenti i cui vettori formeranno un angolo piccolo. Questa sottocategoria di algoritmi introduce due nuovi problemi: l'inserzione di un nuovo utente nel sistema comporta il ricalcolo del modello, e il calcolo del modello stesso è computazionalmente molto pesante, in quanto in ogni sistema il numero di utenti è molto grande (e comunque molto maggiore del numero di *item*).

- ***Item-Based:*** si basano sul concetto di similarità tra *item*, per questo motivo in un sistema con n film, il modello creato avrà dimensione $n \times n$, ed in ogni cella di questa matrice sarà presente il grado di similarità tra il film presente sulla riga e quello presente sulla colonna. Con questa sottocategoria risulta facile effettuare raccomandazioni, in quanto basta moltiplicare il profilo utente (un vettore nello spazio dei film) e il modello creato per avere una lista di *rating* predetti per tutti i film, per il dato utente. Specularmente a quanto succede per gli algoritmi *User-Based*, ogni film sarà rappresentato da un vettore nello spazio degli utenti, e i film simili avranno tra i loro corrispettivi vettori un angolo poco ampio.

2.5 Spiegazione delle raccomandazioni

I sistemi di raccomandazione sono diventati una tecnica popolare per aiutare gli utenti a scegliere i film di loro interesse, evitando loro di doverli cercare

in database caratterizzati da centinaia di migliaia di titoli. Gran parte della ricerca in quest'ambito si basa sullo sviluppo di algoritmi sempre più efficienti nel predire raccomandazioni accurate, trascurando però l'abilità del sistema stesso di spiegare queste raccomandazioni agli utenti.

Tra i primi ad accorgersi dell'importanza delle spiegazioni da dare agli utenti, sono stati Jonathan L. Herlocker, Joseph A. Konstan e John Riedl [9], che nel 2000 cercarono una soluzione al problema, studiando i sistemi di raccomandazioni collaborativi (ACF, *Automated Collaborative Filtering*). Uno svantaggio di questi sistemi consiste nel non essere adatti per raccomandazioni ad alto rischio (come ad esempio la scelta del luogo per le vacanze o per i servizi di investimento) per due motivi: in primo luogo perché sono processi stocastici che calcolano predizioni basate su modelli che sono approssimazioni euristiche di processi umani, e in secondo luogo perché basano i loro calcoli su dati estremamente spari ed incompleti. I sistemi collaborativi funzionano quindi come degli oracoli che danno suggerimenti ma ai quali non è possibile fare domande, ed è per questo che sono usati solo in contesti caratterizzati da basso rischio.

Le spiegazioni sono importanti in quanto aiutano l'utente a stimare l'errore commesso dal sistema di raccomandazione. Esistono due principali tipologie di errori:

1. **Errori di modello/processo:** si presentano quando un utente dà un voto alto a film appartenenti a generi molto diversi tra loro, e lo fa perché quando li ha visti si trovava in contesti distinti (ad esempio una commedia romantica vista con il proprio partner e un film horror visto da solo). Il sistema non riesce a capire questa differenza di situazioni e potrebbe far rientrare l'utente in un cluster di appassionati del genere commedia romantica e continuare a consigliargli film simili.

2. **Errori nei dati:** i dati relativi ad un utente potrebbero essere pochi, di cattiva qualità o caratterizzati da voti ad alta varianza: questo influisce negativamente sulla qualità della raccomandazione.

Inoltre i sistemi collaborativi sono affetti da un altro problema: quando un nuovo film viene inserito nel catalogo, il numero di persone che lo hanno visto è molto ridotto, quindi il sistema si deve basare sui voti di queste poche persone per suggerirlo, sebbene appartengano a cluster differenti.

Costruire quindi un "impianto di spiegazioni" su un sistema di raccomandazioni può portare diversi benefici:

- Giustificazioni: consentono all'utente di capire il ragionamento che sta dietro alla raccomandazione.
- Coinvolgimento dell'utente: permettono all'utente di usare le sue abilità per completare il processo di decisione delle raccomandazioni.
- Istruzione: permettono all'utente di capire le potenzialità e le limitazioni del sistema.
- Accettazione: l'utente accetta i consigli del sistema, in quanto il suo funzionamento gli è noto.

Per creare un sistema di spiegazioni, sono stati fatti due esperimenti. Il primo esperimento ha lo scopo di individuare il modello e le tecniche da usare da supporto alle spiegazioni nei sistemi collaborativi. E' stato effettuato uno studio su alcuni utenti di *MovieLens* (sistema di raccomandazione collaborativo con interfaccia web): a questi utenti è stato consigliato un film, con 21 interfacce per la spiegazione differenti. A questi utenti è stato chiesto di dare un voto, da 1 a 7, sulla possibilità che sarebbero andati a vedere quel film, per ciascuna interfaccia utilizzata. In base ai voti ottenuti da tutti i partecipanti, è stata stilata una classifica delle interfacce che hanno convinto maggiormente l'utente ad andare a vedere quel film. Nelle prime posizioni troviamo:

1. Istogramma con gruppi di utenti: vengono mostrati i voti che utenti con gusti simili hanno dato al film consigliato: i voti sono divisi in tre categorie: voti 1 e 2, voto 3, voto 4 e 5, che corrispondono ad un giudizio negativo, neutro o positivo (figura 2.4).
2. Prestazioni precedenti del sistema di raccomandazione: il sistema mostra il messaggio "*MovieLens* nell'80% dei casi passati ha fatto una predizione corretta per te"
3. Istogramma con i voti dei vicini
4. Tabella con i voti dei vicini

5. Somiglianza con altri film visti. Il sistema mostra il messaggio "Questo film ti è stato suggerito in quanto è simile ad altri film che hai votato 4/5"

6. Presenza di attori/attrici preferiti

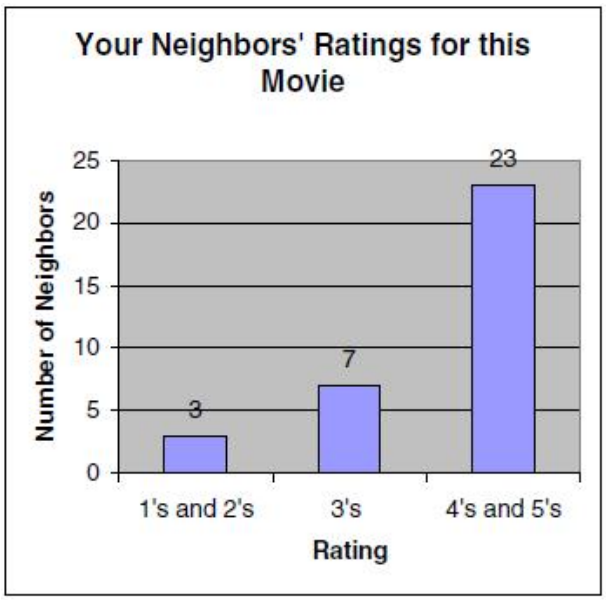


Figura 2.4: Istogramma con i voti dei vicini [9]

Con questo esperimento si è quindi dimostrato che i modi migliori per spiegare le raccomandazioni di un sistema ACF comportano l'uso di istogrammi, un resoconto delle predizioni effettuate nel passato e il paragone dei film con quelli simili già visti e che sono piaciuti. Grazie a queste spiegazioni, è stato possibile mostrare all'utente in modo comprensibile ciò che di sua natura non è ossia il risultato di un modello matematico molto complesso.

Il secondo esperimento ha lo scopo di capire se le spiegazioni possono migliorare l'accettazione dei sistemi ACF e le capacità di filtraggio dei risultati da parte degli utenti. Ai volontari, sempre utenti di *MovieLens*, è stata mostrata una nuova interfaccia per le spiegazioni. Ad ogni soggetto partecipante all'esperimento è stato chiesto di tornare sul sito di *MovieLens* ogni qualvolta avesse visto un nuovo film, compilando un questionario:

- Che film hai visto?

- L'hai visto perché pensavi ti piacesse o per altri motivi?
- Hai consultato *MovieLens* prima di vederlo?
- Se lo hai fatto, qual è stato il consiglio di *MovieLens*?
- Il consiglio di *MovieLens* quanto ha influenzato la tua decisione?
- E' valsa la pena guardare questo film?
- Che voto gli dai?

Un totale di 210 utenti hanno partecipato a questo esperimento, compilando 743 questionari, in 315 dei quali, l'utente ha consultato *MovieLens* prima di vedere il film. In 287 di questi questionari, *MovieLens* ha avuto un qualche effetto sulla decisione dell'utente di vedere o meno il film. In 213 (83%) di questi casi, *MovieLens* non è stata l'unica ragione che ha spinto l'utente a vedere il film. Alla fine dell'esperimento è stato chiesto agli utenti di compilare un questionario facoltativo, che chiedeva se l'interfaccia per le spiegazioni andasse aggiunta all'interfaccia principale del sito e l'86% degli utenti che hanno compilato quest'ultimo questionario ha risposto positivamente. Inoltre è stata data la possibilità a questi utenti di lasciare un commento scritto: la maggior parte dei commenti pervenuti erano positivi e chiedevano la possibilità di vedere i voti espressi dai vicini riguardanti il film raccomandato, in modo da dare consistenza a questi vicini, finora "invisibili", oppure di poter salvare il profilo di questi utenti, in modo da consultarlo in futuro. Il risultato di questo esperimento conferma che le spiegazioni hanno effetto positivo sulle decisioni prese dagli utenti riguardanti la possibilità di guardare o meno un film e le spiegazioni possono migliorare le prestazioni di filtraggio dei risultati.

Mustafa Bilgic e Raymond J. Mooney hanno poi esteso il lavoro di Herlocker in "Explaining Recommendations: Satisfaction vs. Promotion" [3]. Hanno inizialmente identificato i limiti dei sistemi di raccomandazione *Content-Based* e di quelli collaborativi. Come accennato nella sezione 2.4.2 i sistemi collaborativi soffrono di tre principali problemi:

1. Il problema della partenza a freddo (o *Cold-start*): quando si crea un nuovo sistema, non ci sono abbastanza utenti da confrontare.

2. Il problema della sparsità: la matrice URM (*User Rating Matrix*, matrice con i voti degli utenti relativi ai film che hanno visto) è molto sparsa, e ciò rende difficile cercare le correlazioni tra utenti.

3. Il problema del primo *rating*: quando un nuovo film viene inserito nel catalogo nessuno l'ha visto, quindi non sarà consigliato a nessun utente.

D'altra parte, i sistemi *Content-Based* (sezione 2.4.1) soffrono di altri tre problemi:

1. Per certi domini, non ci sono informazioni disponibili riguardanti il contenuto di un *item*, oppure sono molto difficili da analizzare.

2. Formulare gusti non è un compito facile.

3. Questi sistemi suggeriscono film il cui contenuto coincide con quello dei film presenti nel profilo dell'utente. Se un utente ha dei gusti che non sono rappresentati dai film appartenenti al suo profilo, i film di questo genere non gli verranno mai consigliati.

Per ovviare a questi problemi, è stato usato un sistema ibrido, ossia un sistema diviso in tre parti: la prima (*Content Based Ranker*) classifica gli *item* (che nel loro studio sono libri, non film) secondo il grado di corrispondenza tra il loro contenuto e il contenuto dei libri del profilo utente. La seconda parte (il *Rating Translator*) assegna un giudizio a questi libri, in base alla classifica appena stilata. La terza ed ultima parte (*Collaborative filterer*) si occupa di eseguire le raccomandazioni usando una matrice *user-item*. Particolare interessante in questo studio sono le tre tecniche usate per fornire una spiegazione delle raccomandazioni:

1. ***Keyword Style Explanation (KSE):*** approccio usato per spiegare le raccomandazioni dei sistemi *Content-Based*: analizza il contenuto di *item* suggerito e cerca una corrispondenza tra i contenuti degli *item* presenti nel profilo dell'utente. Con questo metodo vengono mostrate le venti parole che hanno avuto più importanza nella raccomandazione. Per ognuna di queste parole è possibile visualizzare un'elenco di libri in cui è presente, ai quali l'utente ha dato un voto alto.

2. ***Neighbor Style Explanation (NSE):*** approccio usato nei sistemi collaborativi: ad un utente che vuole sapere il perché di un certo

suggerimento, viene mostrato un istogramma con i voti che i vicini hanno dato a quell'*item*, divisi in tre categorie: *Bad* (voti 1 e 2), *Neutral* (voto 3) e *Good* (voti 4 e 5). Questo è uno dei ventun sistemi testati da Herlocker, ossia quello che ha dato risultati migliori da un punto di vista della prospettiva promozionale.

3. ***Influence Style Explanation (ISE):*** presenta all'utente una tabella di elementi che hanno avuto un certo peso nella raccomandazione di un determinato *item*. Questa tabella ha tre colonne: nella prima viene mostrato il titolo del libro che l'utente ha letto e votato, nella seconda il voto ricevuto, nella terza il valore di influenza (in centesimi) che quel libro ha avuto sulla raccomandazione. Questo valore è stato calcolato rimuovendo un libro dal profilo utente, calcolando la nuova lista di raccomandazione e misurando la differenza tra i due voti ottenuti dal libro suggerito. Dal momento che il sistema ibrido utilizzato è composto da una parte *Content-Based* e da una collaborativa, questo valore di influenza è stato calcolato separatamente per ciascuna parte, portato in un range [-100, 100], e infine è stata fatta la media aritmetica di questi due valori. A differenza degli altri due metodi, questo è indipendente dal sistema di raccomandazione utilizzato.

Per valutare questi tre metodi per le spiegazioni è stato effettuato uno studio così strutturato su 34 utenti:

1. chiedere agli utenti di votare alcuni libri che hanno letto
2. calcolare una lista di raccomandazione **r**
3. per ogni sistema di spiegazione **e**:
   (a) mostrare **r** all'utente con la spiegazione di **e**
   (b) chiedere all'utente di valutare **r**
4. chiedere all'utente di leggere **r** e di dargli un voto.

Il sistema di spiegazione che minimizza la differenza tra i voti dei punti 3.(b) e 4 sarà quindi il migliore. Per essere sicuri di non influenzare l'utente nella scelta, il titolo e l'autore del libro sono stati nascosti nel punto 3, inoltre gli è stato detto che ogni sistema di spiegazione si riferiva ad un libro diverso. Dato che lo studio così effettuato avrebbe richiesto troppo tempo

per la lettura dei libri (nel punto 4), agli utenti è stata fatta leggere la pagina di *Amazon* relativa a quel libro.

Tipo	μ	σ
Attuale	3.75	1.02
ISE	3.75	1.07
KSE	3.75	0.98
NSE	4.49	0.64

Tabella 2.1: Media e deviazione standard dei voti

Dai risultati (tabella 2.1) è emerso che ***NSE*** sovrastima il voto di 0.74 rispetto alla media. Essendo la media dei voti 3.75 (su 5), una sovrastima di 0.74 è da considerarsi alta. Questa sovrastima inoltre potrebbe portare ad una mancanza di fiducia nel sistema di raccomandazione da parte dell'utente, che potrebbe addirittura decidere di non farne più uso. Dai risultati emerge quindi che ***KSE*** e ***ISE*** sono significativamente più efficienti nel dare spiegazioni agli utenti.

Nava Tintarev e Judith Masthoff [21] [22] [23] [24] invece distinguono i sette scopi delle spiegazioni:

1. Trasparenza: intesa come la consapevolezza da parte dell'utente di come funziona il sistema di raccomandazione: una spiegazione dovrebbe infatti chiarire come sono stati scelti gli *item* che gli sono stati consigliati. E' necessaria una distinzione tra trasparenza e giustificazione: la prima dovrebbe dare un resoconto di come la raccomandazione è stata scelta e di come funziona il sistema, mentre le seconda può essere disaccoppiata dall'algoritmo usato.

2. Scrutabilità: l'utente deve essere in grado di dire al sistema che la raccomandazione che sta facendo è sbagliata. Questo criterio è simile

al precedente (trasparenza), ma differisce nel fatto che la scrutabilità permette all'utente non solo di vedere che voto dato in passato ha influito sulla raccomandazione, ma anche di modificarlo in modo da correggere eventuali errori.

3. Fiducia: è anch'essa legata al criterio di trasparenza e alla possibile interazione con il sistema da parte dell'utente, ma potrebbe anche essere legata all'accuratezza del sistema di raccomandazione. Il modo in cui il sistema può guadagnarsi la fiducia dell'utente è, ad esempio, quello di essere sincero ed ammettere quando non è molto confidente della raccomandazione fatta. Un altro fattore che influisce sulla fiducia è il design del sito web del sistema di raccomandazione: il 46% degli utenti intervistati ha ammesso di dare molta importanza al *layout*, alla tipografia, ai colori ed ai caratteri usati.

4. Persuasione: ossia convincere un utente a provare (o comprare) qualcosa. Questo criterio si basa ancora una volta sullo studio di Herlocker, nel quale era stato scoperto che il metodo più persuasivo per le spiegazioni fosse mostrare all'utente l'istogramma con i voti del film suggeritogli dati dai suoi vicini. Un altro modo per misurare il grado di persuasione di un sistema di raccomandazione consiste nel calcolare la differenza tra il *rating* dato ad un film e il *rating* dato sempre allo stesso film, ma mostrato all'utente con un'interfaccia per le spiegazioni.

5. Efficacia: aiutare gli utenti a prendere buone decisioni. Oltre a convincere un utente a comprare un prodotto, un sistema di raccomandazione deve anche essere in grado di assisterlo permettendogli di prendere decisioni migliori. L'efficacia è molto legata all'accuratezza dell'algoritmo usato per le raccomandazioni: una spiegazione efficace deve essere in grado di aiutare l'utente a capire la bontà dei consigli ricevuti, confrontandoli con le sue preferenze. Un ulteriore modo per misurare l'efficacia è testare lo stesso sistema con e senza la parte relativa alle spiegazioni e vedere se l'utente guarda film che sono più adatti ai suoi gusti.

6. Efficienza: le spiegazioni consentono all'utente di scegliere quale film vedere (tra i consigliati) in minor tempo. L'efficienza è molto importante nei sistemi di raccomandazione *Conversational* [16], che richiedono una continua interazione con l'utente al fine di affinare i suoi gusti

e le sue preferenze: in questo caso può quindi essere calcolata come il numero di interazioni utente-sistema necessarie a raggiungere un *item* soddisfacente. Un'altra metrica per il calcolo dell'efficienza è il numero di modifiche da apportare al profilo al fine di evitare raccomandazioni non adatte

7. Soddisfazione: l'uso del sistema deve essere piacevole per l'utente. Il grado di soddisfazione lo si può chiedere direttamente all'utente, oppure lo si può misurare indirettamente attraverso la fedeltà di quest'ultimo verso il sistema. La soddisfazione differisce dalla persuasione in quanto la prima si riferisce al processo di raccomandazione, mentre la seconda all'*item* raccomandato.

Dopo aver individuato questi sette scopi, hanno dato una descrizione degli stili usati per le spiegazioni. Nella descrizione che segue è usata la seguente notazione:

- **U** è l'insieme di utenti le cui preferenze sono note.
- $\mathbf{u} \in \mathbf{U}$ è l'utente a cui viene fatta la raccomandazione.
- **I** è l'insieme di *item* che possono essere raccomandati.
- $\mathbf{i} \in \mathbf{I}$ è l'*item* di cui dobbiamo predire la preferenza per l'utente **u**.

I quattro stili identificati sono i seguenti:

1. ***Collaborative-Based Style Explanation:*** l'input del sistema di raccomandazione sono i voti di **u** riguardanti *items* di **I**. Questi voti sono usati per identificare utenti con gusti simili a quelli di **u**, i cosiddetti "vicini". Questo stile di spiegazione è usato da *Amazon*: "Chi ha comprato questo oggetto ha comprato anche ...". Questa spiegazione presuppone l'uso di un modello in cui i voti sono implicitamente dedotti dagli acquisti fatti dall'utente. Un sistema efficace per mostrare all'utente i voti che i vicini hanno dato ad un certo oggetto che gli è suggerito di comprare, è sempre quello identificato da Herlocker (istogramma con i voti divisi in tre categorie: *Bad, Neutral* e *Good*)

2. ***Content-Based Style Explanation:*** l'input del sistema di raccomandazione sono i voti di **u** riguardanti un sottoinsieme di *items* presenti in **I**. Questi voti sono usati per creare un classificatore che cerca

una corrispondenza tra i gusti di **u** e i contenuti di **i**. Essendo i sistemi di raccomandazioni *Content-Based* basati sulle proprietà degli *item*, anche lo stile di spiegazioni sarà basato su queste proprietà e dirà all'utente di aver calcolato la similarità tra film basandosi sui contenuti dei film a cui ha dato un voto alto (ad esempio suggeriranno film i cui attori sono presenti in film che gli sono piaciuti). Un approccio più indipendente dal dominio consiste nell'usare la relazione tra *tag* e *item* (*tag relevance*) e la relazione tra i *tag* e gli utenti (*tag preference*) per fare le raccomandazioni: la *tag preference* identifica quanto ad un utente possa interessare un determinato *tag*, mentre la *tag relevance* indica quanto un *tag* appartenga ad un film. Un esempio del funzionamento, usato da *Movielens*, è mostrato in figura 2.5

Figura 2.5: Spiegazione con tag preference e tag relevance, ordinata in base alla tag relevance [24]

3. ***Knowledge and Utility-Based Style Explanation:*** è necessaria la descrizione dei bisogni e degli interessi di **u**, quindi il sistema calcola la corrispondenza tra questi interessi e **i**. Se ad esempio un utente sta cercando una fotocamera digitale, e le sue necessità sono spendere poco, avere poca memoria a disposizione e una bassa risoluzione dell'apparecchio, allora il sistema cercherà tra le fotocamere quella che corrisponde a questi criteri di scelta. Il sistema è inoltre in grado di dire in cosa l'oggetto proposto differisce dalle pretese dell'utente.

4. ***Demographic Style Explanation:*** sono necessarie le informazioni demografiche di **u**: in base a queste informazioni, il sistema trova

utenti simili ad **u**, quindi cerca **i** tra quelli votati dai vicini di **u** con un voto alto. La raccomandazione in questo caso tiene conto sia di quanto i vicini siamo simili ad **u**, sia di come essi abbiano valutato **i**. Questo stile di spiegazione non è molto usato in quanto gli utenti difficilmente vogliono che sia raccomandato loro un oggetto in base al loro genere, età od etnia.

Vig, Sen e Riedl [26] hanno identificato un'entita intermediaria nei sistemi di spiegazione, necessaria per mettere in relazione l'utente con il film consigliato. Come mostrato in figura 2.6, le spiegazioni ricadono in una delle 3 categorie: *item-based*, *user-based* e *feature-based*.

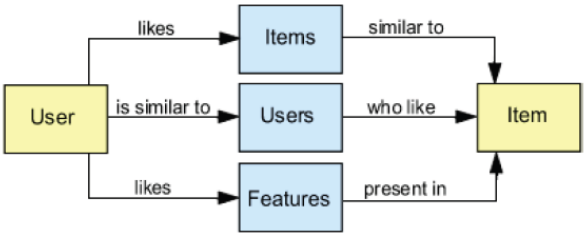


Figura 2.6: Entità intermediarie che mettono in relazione un utente con l'item che gli è stato suggerito [26].

- Spiegazioni *item-based*: l'entità intermediaria è composta da un insieme di *item*, che nel caso specifico sono i film. All'utente è consigliato un film in quanto simile ad altri film che gli sono piaciuti.
- Spiegazioni *user-based*: l'entità intermediaria in questo caso è composta da utenti con i gusti simili a quelli dell'utente al quale fare la raccomandazione. Il film consigliato quindi sarà un film piaciuto a questi utenti.
- Spiegazioni *features-based*: i *tag* sono gli elementi dell'entità intermediaria: all'utente viene suggerito un film contenente *tag* (ad esempio il nome del protagonista, il regista, il genere, ...), che sono presenti in film che gli sono piaciuti.

Esistono due aspetti legati alla spiegazione tramite *tag*, o *tagsplanation*: la relazione tra il *tag* e l'*item* consigliato, chiamata *tag relevance*, e la relazione tra l'utente e il *tag*, ossia *tag preference*. La prima ci dice quanto un *tag* rappresenti un *item*: una possibile misura di questa relazione è data dalla popolarità del *tag*, un'altra è la correlazione tra la preferenza verso un *tag* e la preferenza verso un *item*, ossia capire quanto possa piacere un film ad un utente, sapendo che gli piace il *tag* associato. La *tag relevance* può essere espressa sia come relazione binaria (rilevante o non rilevante), sia come valore compreso in un intervallo: la relazione binaria sicuramente comporta l'uso di un modello concettualmente più facile, ma d'altra parte non soddisfa le esigenze degli utenti che vogliono sapere quanto un *tag* sia rilevante. La *tag preference* invece esprime il grado di interesse dell'utente verso quel *tag*: questo interesse può essere espresso esplicitamente dall'utente, oppure può essere capito dalle sue azioni, ossia in modo implicito, per esempio se un utente dà voti alti a film che hanno un determinato *tag* in comune, il sistema assocerà a quell'utente la preferenza per quel *tag*. Per questo studio è stato usato il dataset di *MovieLens*, che usa un algoritmo *item-item* basato sui k vicini con gusti simili (i *tag* sono stati usati solo in fase di spiegazione e non in fase di raccomandazione). Per prima cosa è stata definita la *tagshare* come la misura della frequenza di un *tag*, usata per assegnare un peso ai film: la *tagshare* di un *tag* t applicata ad un film i è data dal numero di volte che t è stato applicato ad i diviso per il numero di volte che un qualsiasi *tag* è stato associato ad i. Per stimare la *tag preference* di un utente u verso il *tag* t è stata usata la formula seguente:

$$tag\_pref(u,t) = \frac{\left(\sum_{i \in I_u} r_{u,i} \cdot \ tag\_share(t,i)\ \right) + \bar{r}_u \cdot k}{\left(\sum_{i \in I_u} \ tag\_share(t,i)\ \right) + k} \tag{2.1}$$

che risulta indefinita per gli utenti u che non hanno votato nessun film con *tag t*. I_u è l'insieme di film votati dall'utente u, $r_{u,i}$ il voto che l'utente u ha dato al film i e $\bar{r}_u$ la media dei voti dati dall'utente u, k è una costante che vale 0.05 che serve per gli utenti che hanno votato pochi film con *tag t*. Per quanto riguarda la *tag relevance*, è stata usata la seguente formula:

$$tag\_rel(t,i) = \begin{cases} \text{pearson(X,Y)} & \text{se } t \text{ è presente in } i \\ 0 & \text{altrimenti} \end{cases} \tag{2.2}$$

essendo U_{ti} l'insieme di utenti che hanno votato i e hanno una *tag preference* definita per t (e lo è se hanno votato almeno un film con tag t). X è l'insieme

dei voti del film i espressi dagli utenti in U_{ti} ai quali viene sottratta la media dei voti dell'utente, ossia $X = \{r_{u,i} - \bar{r}_u : u \in U_{ti}\}$. Y è l'insieme dei valori di *tag preference* del *tag* t per tutti gli utenti in U_{ti}, al quale viene sottratto la media dei voti dell'utente, ossia $Y = \{tag\_pref(u,t) - \bar{r}_u : u \in U_{ti}\}$. E' stato quindi effettuato un sondaggio su 556 utenti di *MovieLens*, ad ognuno dei quali è stata mostrata una *tagsplanation* per quattro diversi film ed è stato chiesto di valutare i seguenti punti con una scala di valori da 1 a 5:

1. Questa spiegazioni mi aiuta a capire il voto che è stato predetto per questo film.
2. Questa spiegazione mi aiuta a capire quanto mi possa piacere questo film.
3. Questa spiegazione mi aiuta a capire se questo film è adatto al mio umore attuale.

Inoltre ad ogni utente è stato mostrato un *tag* usato nella spiegazione di un film che gli è stato suggerito, e gli è stato chiesto di valutare (sempre usando una scala da 1 a 5) queste tre affermazioni.

1. Questo elemento mi aiuta a capire il voto che è stato predetto per questo film.
2. Questo elemento mi aiuta a capire quanto mi possa piacere questo film.
3. Questo elemento mi aiuta a capire se questo film è adatto al mio umore attuale.

Infine sono stati mostrati ad ogni utente due film che ha visto e votato in passato, con la relativa *tagsplanation* e gli è stato chiesto di dare un voto da 1 a 5 alla seguente affermazione:

- In generale questa è una buona spiegazione data la mia conoscenza del film.

I risultati ottenuti confermano che sia la *tag relevance* che la *tag preference* coprono un ruolo importante nella *tagsplanation* ed aiutano a raggiungere gli obiettivi di giustificazione ed efficacia già introdotti da Tintarev (et al), oltre a capire se un film è adatto oppure no in base all'umore dell'utente, infatti oltre l'80% dei partecipanti allo studio hanno espresso un'opinione favorevole riguardante l'uso dell'interfaccia grafica delle *tagsplanation*.

2.6 Confidenza delle raccomandazioni

I sistemi di raccomandazione sono nati per ovviare al problema del sovraccarico di informazioni presenti, consentendo all'utente di poter scegliere l'oggetto di suo interesse tra un sottogruppo molto ristretto della totalità di *items*: è quindi di notevole importanza la confidenza con la quale un sistema di raccomandazione consiglia film agli utenti. Adomavicius, Kamireddy e Kwon hanno affrontato il problema della confidenza introducendo un metodo di filtraggio delle raccomandazioni basato sulla varianza dei voti dei film [1]: hanno infatti notato che all'aumentare di questa varianza, l'accuratezza dei consigli diminuisce. Per i loro test hanno usato un dataset di film, hanno diviso i voti in alti (4 e 5) e bassi (1, 2 e 3) e hanno valutato l'accuratezza della raccomandazione come percentuale di film con voti alti presente nella TopN, attraverso il calcolo della *Precision-in-top-N*. Sono stati proposti tre approcci:

1. ***Simple Filtering Approach:*** solitamente, in una raccomandazione, vengono predetti i voti che un utente darebbe ai film in catalogo, quindi si ordinano e si consigliano all'utente i primi N film della lista. Con questo approccio, i film sono stati filtrati in due modi: inizialmente sono stati tolti dall'elenco i film il cui voto predetto non superava una certa soglia (soglia accettabile, nel loro studio fissata a 3.5 su 5), quindi i film rimasti sono stati filtrati in base al valore della deviazione standard dei loro voti, tenendo solo i film con questo valore minore di D. Essendo la deviazione standard media dei voti del database molto prossima ad 1, è stato deciso di considerare valori di D compresi tra 0.8 e 1.2, con incrementi di 0.05. In questo modo vengono consigliati all'utente solo film con voto predetto alto e varianza dei loro voti bassa. Dal momento che una raccomandazione non deve essere solo accurata ma anche utile, è stata usata anche la metrica di copertura per valutarla, calcolata come numero di film diversi raccomandati a tutti gli utenti. Questa metrica risulta in contrasto con quella della *precision* ed è quindi necessario trovare un trade-off tra i due valori, in modo da avere una raccomandazione sia accurata, sia che non consigli a tutti gli utenti gli stessi film. Per questo motivo sono stati proposti gli altri due approcci.

2. ***Smart Approach:*** questo approccio usa la deviazione standard per

modificare il voto predetto per i film della lista di raccomandazione: dal voto predetto viene sottratto il valore della deviazione standard, quindi vengono mostrati solo i film che hanno il nuovo voto superiore ad una certa soglia.

3. ***Smart Approach:*** questo approccio consiste in una piccola modifica di quello precedente: dopo aver tolto dal voto predetto il valore di una deviazione standard, si tengono solo i film il cui valore supera la soglia prescelta, quindi si riordina la lista ottenuta in base a questo valore.

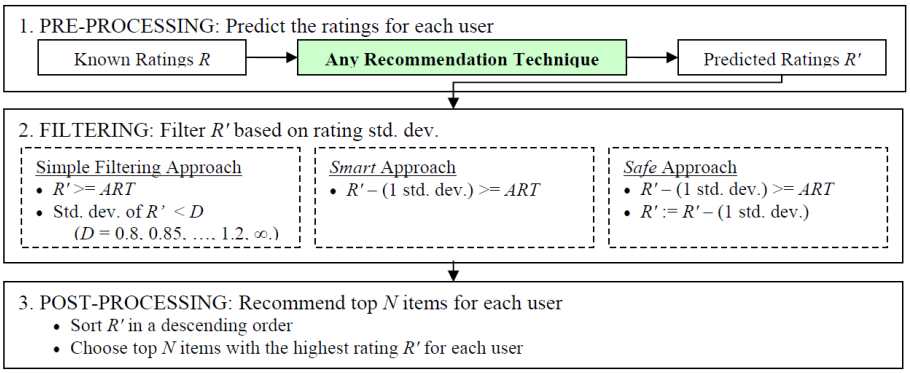


Figura 2.7: Tre filtri basati sulla varianza dei voti. ART: Acceptable Rating Threshold [1]

I tre approcci sono stati testati con una versione modificata del dataset di *MovieLens*: sono infatti stati considerati solo gli utenti che hanno dato un voto ad almeno venti film, e i film che hanno ricevuto almeno venti voti; in questo modo si è ottenuta una matrice di 917 utenti per 937 film, per un totale di 94443 voti e una densità pari all'11%. Questi tre filtri sono indipendenti dall'algoritmo utilizzato di raccomandazione, in quanto modificano la lista ottenuta e non ne influenzano la creazione: per i test è stato scelto l'utilizzo dell'algoritmo Item-Item Cosine kNN. Dai risultati ottenuti emerge che gli approcci *Smart* e *Safe*, rispetto all'approccio si *Simple Filtering*, ottengono una precisione alta in corrispondenza di una copertura anch'essa elevata.

In letteratura è presente un altro studio riguardante la confidenza: secondo McNee, Lam, Guetzlaff, Konstan e Riedl [20] (ricercatori di *GroupLens*) è infatti legata al numero di voti ottenuti da ciascun film. Hanno infatti effettuato un esperimento, calcolando l'errore assoluto medio commesso nelle raccomandazioni di *MovieLens*, dividendo i film consigliati in gruppi di venti: i film con il più alto valore di errore sono quelli che hanno ottenuto meno voti. Sono quindi stati definiti molto rischiosi i film con meno di 40 voti ottenuti (contrassegnati in fase di raccomandazione con l'icona di due dadi), e di medio rischio quelli che ne hanno ottenuti da 41 a 80 (contrassegnati con un dado). Nel dataset di *Movielens*, il 20% di film sono considerati molto rischiosi e un ulteriore 10% è da considerarsi di medio rischio. E' stato effettuato un test per verificare gli effetti della confidenza sul comportamento degli utenti verso il sistema di raccomandazione: gli utenti partecipanti sono stati divisi in due gruppi, quelli che già usavano il sistema di raccomandazione di *MovieLens*, e quelli che non lo avevano mai usato. Dopo un periodo di prova del sistema da parte di questi utenti, sono state poste loro alcune domande riguardanti la loro esperienza con *MovieLens*:

1. Quanto spesso credi che le raccomandazioni di *MovieLens* siano corrette?
2. Pensi veramente di poter utilizzare questo sistema per scegliere i film da guardare?
3. Quanto sei soddisfatto della tua esperienza con *MovieLens*?
4. Hai notato le icone dei dadi?
5. Sai cosa significano quei dadi?
6. Quanto ritieni utile l'icona dei dadi nello scegliere il film da vedere?
7. Hai evitato i film contrassegnati dai dadi?
8. Prima di scegliere i film contrassegnati dai dadi, ti sei informato da altre fonti?
9. Saresti più propenso a cercare informazioni da altre fonti riguardo i film contrassegnati dai dadi, rispetto a quelli che non lo sono?

Da questo test è emerso che mostrare la confidenza aumenta la soddisfazione degli utenti e gli utenti stessi solitamente evitano i film contrassegnati

dai dadi. Inoltre è emerso che gli utenti che già facevano uso di *MovieLens* non fanno molto uso delle icone dei dadi, in quanto hanno già una loro idea sul funzionamento del sistema e su quanto si possano fidare di esso, mentre i nuovi utenti tengono maggiormente in considerazione questa segnalazione del grado di confidenza, soprattutto se si sono informati riguardo al funzionamento delle icone dei dadi.

Capitolo 3

Dataset e algoritmi utilizzati

3.1 Dataset

Per lo studio delle spiegazione e, in seguito, per lo studio della confidenza di quest'ultime, sono stati utilizzati dataset reali, dalle caratteristiche diverse. Per le nostre analisi è stato necessario suddividere gli utenti presenti in questi dataset in tre categorie, in base al numero di film che hanno visto:

- Profilo A: fanno parte di questo profilo gli utenti occasionali del sistema di raccomandazione, ossia quelli che guardano (e votano) film raramente; è stato scelto di assegnare a questo profilo gli utenti che hanno visto da 3 a 5 film: non sono stati presi in considerazione gli utenti che hanno visto meno di tre film in quanto non sarebbero stati adatti allo studio effettuato per le spiegazioni.
- Profilo B: in questo profilo rientrano gli utenti che hanno visto da 6 a 10 film.
- Profilo C: fanno parte di questo profilo gli utenti che hanno visto da 11 a 20 film: è stato deciso, ameno per la prima parte di test, di non andare oltre la soglia dei 20 film in quanto il tempo richiesto per le nostre analisi cresce esponenzialmente con il numero di film visti dall'utente: un aumento anche insignificante di questo valore avrebbe richiesto molto più tempo di esecuzione delle analisi fatte.

Per quanto riguarda gli studi esposti nel capitolo 7, relativi al calcolo di *Recall* e *Fall-out*, è stato necessario fare una distinzione anche sugli *item*:

sono stati definiti popolari quei film che raccolgono il 33% dei *rating* presenti nella matrice URM, e non popolari i film che raccolgono il restante 67% di *rating*.

3.1.1 MovieRec

Uno dei dataset di cui siamo in possesso è privato, quindi verrà chiamato con un nome fittizio: *MovieRec*. Il dataset di *MovieRec* è un dataset binario, ossia contiene *rating* impliciti: il profilo utente è dato da una sequenza di zeri in corrispondenza dei film che l'utente non ha visto e di uni in corrispondenza di film che l'utente ha acquistato tramite il servizio che l'operatore stesso offre di *Video On Demand*. Il dataset contiene informazioni relative a 50889 utenti (la cui distinzione in base al profilo d'appartenenza è mostrata in figura 3.1) e 985 film, 46 dei quali popolari (poco meno del 5% del totale dei film).

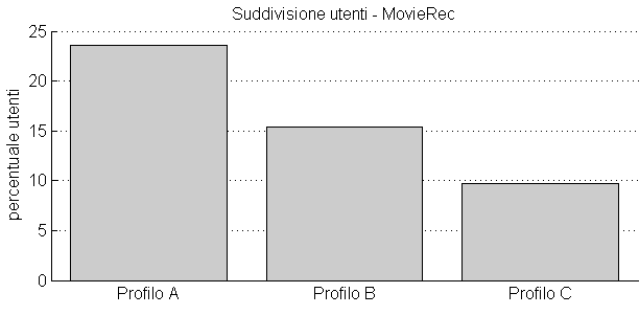


Figura 3.1: Suddivisione degli utenti di MovieRec nei tre profili

3.1.2 Netflix

Netflix è una società americana che noleggia film su DVD e da qualche hanno effettua anche un servizio di streaming per i propri abbonati. Il dataset è di tipo esplicito, con una scala di *rating* compresa tra 1 e 5. Il dataset originale è composto da 480188 utenti e 17770 film, solo il 2% dei quali appartiene alla categoria dei popolari. Essendo questo dataset troppo numeroso per i nostri calcoli (i quali avrebbero impiegato troppo tempo per essere svolti), per i test relativi alle spiegazione e alla confidenza ne è stata usata una versione ridotta, che comprende 126603 utenti e 6890 film. La suddivisione degli

utenti tra le tre categorie (per entrambe le versioni del dataset) è mostrata nella figura 3.2.

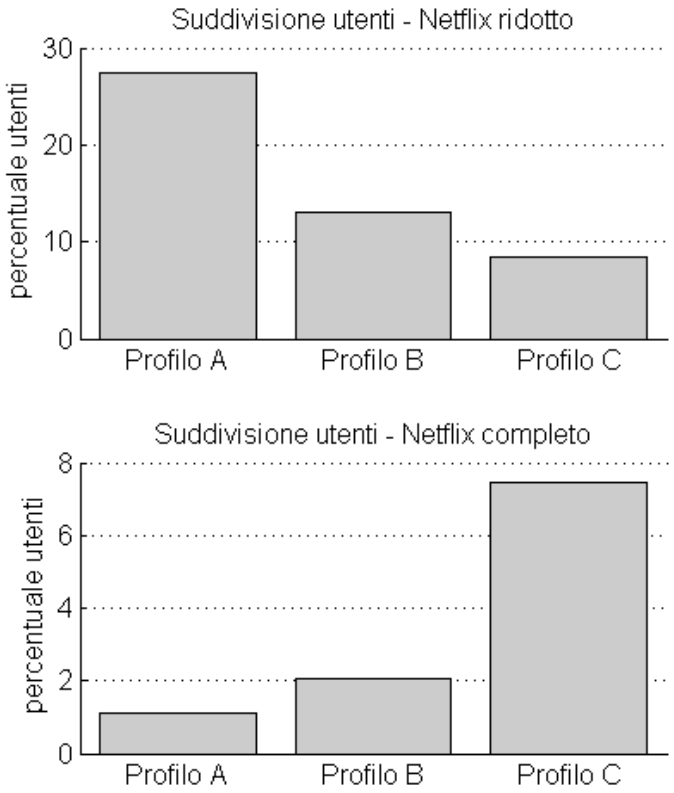


Figura 3.2: Suddivisione degli utenti di Netflix nei tre profili

3.2 Algoritmi utilizzati

In questa sezione verranno mostrati gli algoritmi usati per ottenere le liste di raccomandazione, specificando a quale *dataset* sono applicabili, e gli eventuali parametri di cui necessitano.

3.2.1 Algoritmi con modello nello spazio latente

Per alcuni algoritmi di raccomandazione è possibile definire un parametro che verrà usato in fase di creazione del modello, il parametro di *latent si-*

ze (spazio latente), che serve a ridurre le dimensioni del modello tenendo in considerazione solo le prime *ls features*, ossia caratteristiche nascoste della matrice del dataset. Queste *ls* caratteristiche, ricavate tramite una decomposizione delle matrici del dataset, possono rappresentare:

- una relazione implicita tra gli utenti e gli *item* negli algoritmi collaborativi. In questo caso viene effettuata la decomposizione della matrice URM.
- il contenuto degli *item* negli algoritmi di tipo *Content-Based*, ossia i metadati (capitolo 2.2) della matrice ICM, che in questo caso viene decomposta.

Tramite l'utilizzo delle features è anche possibile ridurre il rumore perché vengono eliminati i dati con meno contenuto informativo a beneficio di quelli più ricchi di informazioni [7]: essendo la matrice URM di dimensione $m \times n$ (con m numero di utenti ed n numero di *item*) e la matrice di ICM di dimensioni $s \times n$ (con s numero di metadati relativi agli n film), con la decomposizione nello spazio latente si possono rappresentare le stesse informazioni relative ad utenti ed *item* in uno spazio ls, con $ls \ll m, n, s$.

La decomposizione usata per ridurre queste matrici nello spazio latente è una tecnica matematica denominata SVD (*Singular Value Decomposition*) [11]: tramite questa tecnica è possibile decomporre qualsiasi matrice M di dimensione $a \times b$ in 3 matrici:

- U**:** matrice unitaria di dimensione $a \times a$
- Σ**:** matrice diagonale di dimensione $a \times b$
- V^T**:** trasposta coniugata di una matrice unitaria di dimensione $b \times b$

I cosiddetti valori singolari sono i valori presenti sulla diagonale della matrice Σ: tramite la riduzione nello spazio latente è possibile tenere solo i primi ls valori singolari, passando questo parametro alla funzione che effettua la decomposizione SVD, riducendo così le dimensioni delle tre matrici, come verrà mostrato nelle sezioni 3.2.5 e 3.2.6.

3.2.2 Item-Item Cosine kNN

L'algoritmo Item-Item Cosine kNN è un algoritmo collaborativo di tipo *Item-Based*, in cui ogni film è rappresentato da un vettore di m elemen-

ti, se m sono gli utenti del sistema, composto quindi dai voti che ha ricevuto (e zeri in corrispondenza degli utenti che non l'hanno visto).

Nella fase *Batch* viene creato il modello dell'algoritmo, prendendo come *input* la matrice URM di dimensione $m \times n$, che consiste in una matrice $n \times n$ ossia la matrice di similarità tra *item*: in posizione *i-j* di questa matrice sarà quindi presente il valore di similarità tra il film i e il film j. Il valore di similarità è dato dal coseno dell'angolo tra i vettori rappresentanti i due film, e vale 1 se si confronta un film con se stesso, mentre vale 0 se non esistono utenti che hanno dato un voto sia al film i che al film j [18]:

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

Chiamando II (*Item-Item*) la matrice di similitudine creata, la lista di raccomandazione per un utente u è calcolata in *Real-time* ed è data da:

$$RecomList = userProfile(u) \times II \tag{3.2}$$

In questo modo ad ogni film i non votato dall'utente u verrà dato un valore composto dalla somma degli indici di similarità che questo film ha con ogni film presente nel profilo. Una volta effettuata questa operazione per ogni film non visto dall'utente, verrà stilata una classifica in base ai valori ottenuti (che nei dataset espliciti corrispondono ad un possibile voto che l'utente darebbe a quell'*item*), quindi si tengono i primi N valori. *kNN* significa *k-Nearest-Neighbor*, e indica che in fase di creazione del modello vengono conservati i valori relativi soltanto ai k film più simili al film i, mentre tutti gli altri valori vengono considerati rumore. Usare un k molto grande può essere un problema in quanto vengono considerati anche *item* che non sono simili ad i, mentre un uso di k troppo piccolo potrebbe portare all'eliminazione di film che hanno un certo peso sull'indice di similitudine tra i e gli altri film. K può essere posto uguale ad infinito: in questo caso tutti i film presenti in catalogo vengono considerati con il loro grado di similitudine rispetto ad i, e non solo i k più simili; l'algoritmo prende quindi il nome di *Item-Item Cosine*.

Questo algoritmo funziona sia con dataset impliciti che con dataset espliciti: è stato quindi usato con il dataset di *Netflix* e con quello di *MovieRec*, con un valore di k pari a 100.

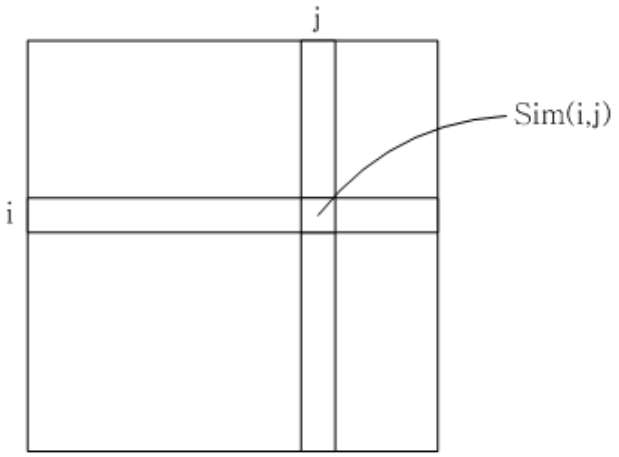


Figura 3.3: Matrice di similitudine Item-Item Cosine kNN

3.2.3 Item-Item Pearson

L'algoritmo Item-Item Pearson è di tipo collaborativo *Item-Based*. In fase *Batch* viene creato il modello ricevendo come input la matrice URM, quindi il sistema si occupa di creare la matrice di similitudine tra gli *n item* presenti. In questo caso la matrice di similitudine viene creata calcolando il coefficiente di correlazione di Pearson tra i diversi film: chiamando U l'insieme degli utenti che hanno dato un voto sia ad i che a j, l'indice di similitudine tra i e j si calcola in questo modo [18]:

$$sim(i,j) = \frac{\sum_{u \in U} \left(R_{u,i} - \bar{R}_i\right)\left(R_{u,j} - \bar{R}_j\right)}{\sqrt{\sum_{u \in U} \left(R_{u,i} - \bar{R}_i\right)^2}\sqrt{\sum_{u \in U} \left(R_{u,j} - \bar{R}_j\right)^2}} \tag{3.3}$$

dove $\bar{R}_i$ è la media dei voti ricevuti dal film i e $R_{u,i}$ è il voto che l'utente u ha dato al film i. Una volta calcolata questa matrice, viene predetto il voto per ogni singolo film i che l'utente u non ha visto effettuando una somma pesata dei voti che u ha dato ai film simili ad i:

$$P_{u,i} = \frac{\sum_{tutti\ i\ film\ simili,\ N} (s_{i,N} * R_{u,N})}{\sum_{tutti\ i\ film\ simili,\ N} (\mid s_{i,N} \mid)} \tag{3.4}$$

Il valore $P_{u,i}$ ottenuto è da considerarsi come voto predetto per il film i, quindi la lista di raccomandazione si ottiene ordinando questi valori in modo decrescente e tenendo soltanto i primi N valori. Così come per l'algoritmo

Item-Item Cosine kNN, anche con questo algoritmo è possibile passare in input alla funzione che crea il modello il parametro *kNN*, in modo che la similitudine tra item venga calcolata solo con i primi k film più simili: il valore di k usato in questi test è 100. Questo algoritmo funziona solo con dataset espliciti, in quanto nel calcolo del coefficiente di correlazione di Pearson è necessario avere la media dei voti ricevuti da ciascun film: è quindi stato usato con il dataset di *Netflix*.

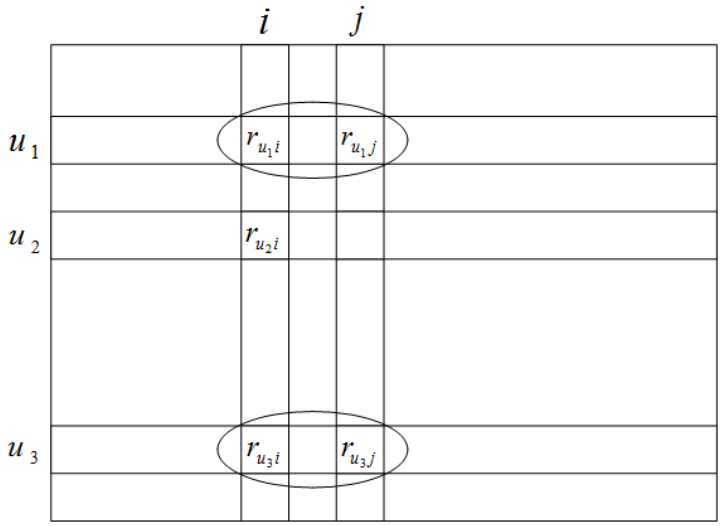


Figura 3.4: Matrice di similitudine tra item: il coefficiente $sim(i, j)$ viene calcolato isolando gli utenti che hanno visto entrambi i film.

3.2.4 Asymmetric SVD

Asymmetric SVD è un algoritmo collaborativo di tipo *Item-Based* ideato da Yehuda Koren [14][15] in occasione del concorso *Netflix Prize* e che gli è valso il primo premio. Durante la fase *Batch* viene creato il modello, composto da diversi vettori e matrici che verranno usate per calcolare il *rating* predetto per il film i dall'utente u [14][15]:

$$r_{ui} = b_{ui} + q_i^T \left(\mid R(u) \mid^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + \mid N(u) \mid^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \tag{3.5}$$

dove:

- $b_{ui} = \mu + b_i + b_u$, con μ media globale dei *rating* presenti nel dataset, b_u rappresenta la distanza (bias) da μ dei *rating* dati dall'utente u ai film che ha visto e b_i è la distanza da μ della media dei voti ricevuti dall'*item i*.
- $\mid R(u) \mid$ è la cardinalità dei *rating* espliciti dati da u.
- $\mid N(u) \mid$ è la cardinalità dei *rating* impliciti dati da u.
- q, x, y sono tre vettori che rappresentano rispettivamente gli *item* non votati da u, gli *item* a cui u ha dato un voto espliciti e gli *item* a cui u ha dato un voto implicito. Tutti i vettori rappresentano film nello spazio latente di dimensione ls.

Da questa formula è stato possibile ricavare la matrice di similarità tra item:

$$SIM = \left(Q^T \cdot Z\right)^T \tag{3.6}$$

- Q è data dalla concatenazione di tutti i vettori colonna q_i, rappresentanti i film non visti da u, ed ha quindi dimensione $ls \times n$, con n numero film film presenti nel dataset.
- Z è la concatenazione di Z_1, Z_2, ..., Z_N ognuno rappresentante un film visto da u e quindi composto da $(r_{uj} - b_{uj})x_j + y_j$, e quindi di dimensione $ls \times N$.

Il prodotto $(Q^T \cdot Z)$ viene trasposto per avere una matrice di similitudine che abbia i film visti da u presenti sulle righe: questa è solo una convenzione adottata da noi in quanto abbiamo sempre considerato i film visti presenti sulle righe, mentre i film suggeriti presenti sulle colonne. Questa matrice di similitudine, per come è stata costruita, è diversa per ogni utente e di conseguenza molto ridotta (non sono riportati tutti film sulle righe ma solo quelli visti dall'utente). Nel creare quest'algoritmo, Koren ha settato diversi parametri in funzione della matrice URM da lui usata, ossia quella di *Netflix* completa: nell'eseguire i nostri test con la matrice ridotta, ci siamo accorti che durante la creazione del modello si faceva *overfitting* sui bias di utenti e di film: ne conseguiva che ad ogni utente venivano consigliati gli stessi film. Per rimediare a questo inconveniente è stato necessario modificare alcuni parametri ed eseguire diversi test prima di poter proseguire con il corretto uso dell'algoritmo, sul solo dataset di *Netflix* (in quanto necessita di *rating* espliciti). Il parametro *ls* è stato posto uguale a 200.

3.2.5 Pure SVD

Pure SVD è un algoritmo collaborativo di tipo *Item-Based* che utilizza la decomposizione SVD nello spazio latente di dimensioni *ls* della matrice URM, tenendo quindi solo i primi *ls* valori singolari [18]:

$$\left[U, \Sigma, V^T\right] = svds(URM, ls) \tag{3.7}$$

La matrice URM, di dimensione $m \times n$, viene scomposta in tre matrici (figura 3.5):

- U è una matrice unitaria di dimensioni $m \times ls$
- Σ è una matrice diagonale di dimensioni $ls \times ls$
- V^T è la trasposta coniugata di una matrice unitaria di dimensioni $ls \times n$

Una volta effettuata questa decomposizione, il modello dell'algoritmo calcolato in fase *Batch* è dato dalla matrice V, che risulta essere di dimensioni molto inferiori rispetto alla matrice URM di partenza, ma anche rispetto al modello creato con altri algoritmi di tipo *Item-Based* visti in precedenza. Per ricavare la matrice di similitudine tra *item* basta moltiplicare $V \times V^T$, ottenendo così una matrice di dimensione $n \times n$ con *item* presenti sia sulle righe che sulle colonne. In fase *Real-time* viene effettuata la raccomandazione per l'utente u nel seguente modo:

$$RecomList \ = userProfile(u) \times V \times V^T \tag{3.8}$$

Questa lista verrà poi ordinata in modo decrescente e verranno mostrati all'utente i primi N film. Il parametro della dimensione latente usato nei nostri test è pari a 200. Con questo algoritmo sono stati usati entrambi i dataset in nostro possesso, in quanto funziona sia con *rating* impliciti che con *rating* espliciti.

3.2.6 LSA Cosine

LSA Cosine (*Latent Semantic Analysis*) è un algoritmo, a differenza di quelli precedenti, di tipo *Content-Based*: utilizza quindi i contenuti dei film (che ricordiamo possono essere parole relative al titolo, l'autore, il regista, ...) per stabilire relazioni si similarità tra gli *item* presenti nel profilo dell'utente

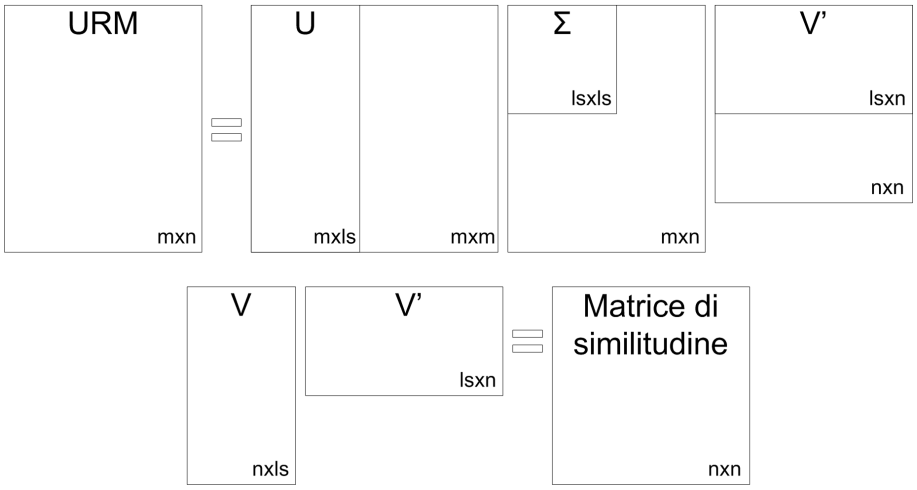


Figura 3.5: Creazione modello e matrice di similitudine Pure SVD

e quelli da suggerire in fase di raccomandazione. Per la creazione del modello è quindi necessaria la matrice ICM (*Item Content Matrix*) contenente l'elenco dei film presenti a catalogo e per ognuno il valore del peso che un certo contenuto (*stem*) ha per quel film; questo peso dipende sia dal numero di volte che un certo *stem* compare nella descrizione del film (più volte compare, maggiore è il peso), sia da quante volte lo stesso *stem* compare nelle descrizioni di tutti gli altri film (più volte compare, più basso sarà il peso). In fase *Batch* viene creato il modello tramite la decomposizione SVD [11], ma al contrario di quanto avviene per l'algoritmo Pure SVD, la decomposizione viene effettuata sulla matrice ICM e non sulla URM (figura 3.6):

$$\left[U, \Sigma, V^T\right] = svds(ICM, ls) \tag{3.9}$$

Il modello in questo caso è dato da $\Sigma \times V^T$, mentre la matrice di similitudine (SIM) è data da:

$$SIM = (\Sigma \times V^T)^T (\Sigma \times V^T) \tag{3.10}$$

La lista di raccomandazione è come al solito creata in *Real-time* quando l'utente la richiede: viene calcolato il coseno dell'angolo presente tra il vettore utente $\vec{u}$ (che rappresenta i gusti dell'utente) e gli *item* $\vec{j}$ che l'utente non ha votato: più quest'angolo è piccolo, più il valore del coseno (ossia l'indice di similitudine) è alto. In base a questo valore viene creata la lista di racco-

mandazione TopN, tenendo gli N film che più di tutti dovrebbero soddisfare i gusti dell'utente.

Questo algoritmo è stato usato soltanto con il dataset di *MovieRec*, in quanto non siamo in possesso della matrice ICM riguardante il dataset di *Netflix*. Il parametro *ls* anche in questo caso è stato posto uguale a 200.

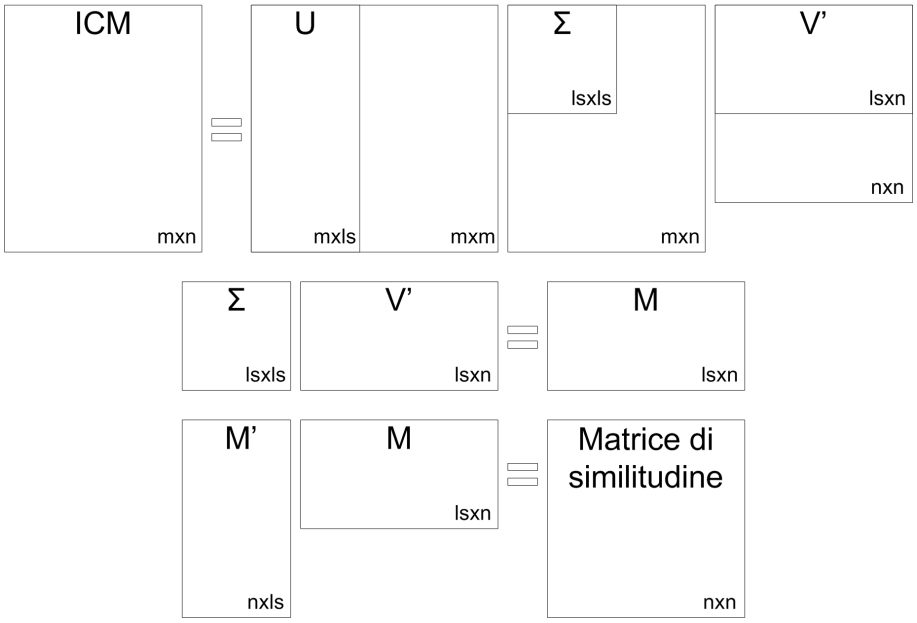


Figura 3.6: Creazione modello e matrice di similitudine LSA Cosine

Nei nostri studi non sono stati usati algoritmi base, come ad esempio l'algoritmo Top-Rated, in quanto le nostre analisi si basano sulla modifica del profilo dell'utente e la conseguenza che ciò comporta sulla lista di raccomandazione: Top-Rated è un algoritmo che consiglia a tutti gli utenti la stessa lista TopN, rappresentante i film più votati presenti in catalogo: non prendendo in input nessun dato i risultati che si sarebbero ottenuti modificando il profilo utente non sarebbero stati diversi da quelli ottenuti con il profilo originale.

Nella tabella 3.1 sono riassunte le informazioni relative ai tipi di algoritmo che è possibile usare con i diversi *dataset* a nostra disposizione.

Tipo	**Nome**	**MovieRec**	**Netflix**
Collaborativo (Item-Based)	Cosine kNN	X	X
Collaborativo (Item-Based)	Pearson		X
Collaborativo (spazio latente)	Asymmetric SVD		X
Collaborativo (spazio latente)	Pure SVD	X	X
Content-Based	LSA Cosine	X	

Tabella 3.1: Tabella riassuntiva dei vari algoritmi che è possibile usare con i dataset a nostra disposizione

Capitolo 4

Perturbazione dei rating

Come accennato nel capito 1, il nostro lavoro si è basato sulla modifica del profilo degli utenti per cercare di dare una spiegazione ai film raccomandati e per cercare anche di dargli un valore di confidenza, ossia di certezza (da parte dell'algoritmo usato) che quel determinato film sia d'interesse per l'utente analizzato.

In questo capitolo verranno mostrati gli effetti della perturbazione sulla lista TopN di raccomandazione [25]. Per perturbazione si intende una modifica, naturale o forzata, del modello o del profilo dell'utente. In tabella 4.1 sono mostrati i possibili casi di perturbazione e i loro effetti:

	Profilo utente	**Modello**
Naturale	eccesso di *time-diversity*	eccesso di *time-diversity*
	assenza di *time-diversity*	assenza di *time-diversity*
Forzato	spiegazioni confidenza	attacco
		privacy

Tabella 4.1: Effetti della perturbazione dei rating nei sistemi di raccomandazione [25]

L'architettura di un sistema di raccomandazione, come mostrato nel capitolo 2.3, è composta da una parte *Batch* che crea il modello partendo dalle matrici URM e ICM e da una parte *Real-time*. Il modello M risulta quindi una descrizione compatta delle due matrici:

$$\mathrm{M} = f(\mathrm{URM}(t), \mathrm{ICM}(t)) \tag{4.1}$$

Questo tipo di architettura risulta scalabile, in quanto il calcolo del modello è effettuato *off-line* dal processo *Batch*, e permette di fare raccomandazioni per tutti gli utenti anche se non è aggiornato con gli ultimi *rating* e i nuovi utenti. Siccome il calcolo del modello è computazionalmente pesante e richiede diverso tempo per essere svolto, solitamente si effettua ad intervalli di tempo regolari e non troppo distanti tra loro (solitamente ogni settimana). La lista di raccomandazione TopN (che chiameremo L) quindi è basata su un modello vecchio:

$$\mathrm{L}(t) = g(\mathrm{M}(t - \Delta t), \mathrm{P}(t)) \tag{4.2}$$

ossia è funzione dell'ultimo modello creato e del profilo utente attuale, con Δt intervallo di tempo dall'ultimo aggiornamento del modello.

La perturbazione dei voti può essere quindi effettuata sul profilo utente P oppure sulla matrice URM, influenzando così il modello M; può inoltre essere classificata come naturale o forzata.

4.1 Perturbazione naturale

La perturbazione naturale è collegata all'evoluzione temporale del profilo utente e della matrice URM. Dato un generico istante di tempo t, definiamo:

- $\mathrm{I}(t)$ come l'insieme di *item* attivi al tempo t, ossia quegli *item* che hanno ricevuto almeno un voto da parte degli utenti.

- $\mathrm{U}(t)$ come l'insieme di utenti attivi al tempo t, ossia quegli utenti che hanno dato almeno un voto ad un film.

Se consideriamo due istanti di tempo distinti, t_0 e t_1, con $t_0 < t_1$, la matrice URM al tempo t può essere scomposta in 4 matrici che descrivono la sua evoluzione nell'intervallo di tempo $[t_0, t_1]$:

1. $\mathrm{A}(t_1 \mid t_0)$ contenente i voti collezionati fino al tempo t_1 riguardanti gli utenti e i film attivi al tempo t_0

2. $\mathrm{B}(t_1 \mid t_0)$ contenente i voti collezionati nell'intervallo di tempo $[t_0, t_1]$, dati dagli utenti attivi in t_0 ai film non attivi in t_0

3. $\mathrm{C}(t_1 \mid t_0)$ contenente i voti collezionati nell'intervallo di tempo $[t_0, t_1]$, dati dagli utenti non attivi in t_0 ai film attivi in t_0

4. D($t_1 \mid t_0$) contenente i voti collezionati nell'intervallo di tempo $[t_0, t_1]$, riguardanti gli utenti e i film non attivi al tempo t_0

La diversità della lista di raccomandazione nel tempo è quindi dovuta ad un cambiamento del modello o del profilo utente, o di entrambi: il modello può cambiare in seguito all'inserimento di nuovi voti, di nuovi utenti o di nuovi film nel catalogo del sistema, mentre il profilo utente può cambiare in seguito all'inserimento di un voto da parte dell'utente stesso.

E' necessario evitare sia l'assenza che l'eccesso di perturbazione naturale dei *rating*: l'assenza comporterebbe la staticità del sistema, il che comporterebbe ad avere liste di raccomandazioni che consigliano sempre gli stessi film, mentre un eccesso comporterebbe un dinamicità delle liste di raccomandazione che potrebbe infastidire l'utente, il quale si vede suggerire dal sistema film diversi ogni volta. Questa dinamicità è particolarmente evidente quando un utente ha visto pochi film e aggiunge un nuovo *rating*, oppure quando accede al servizio per usufruire della raccomandazione ad intervalli di tempo distanti.

La perturbazione naturale può essere analizzata valutando quanto L cambi nel tempo, attraverso la *diversity* e la *novelty*: la *diversity* misura la diversità tra due liste di raccomandazione TopN, e si calcola come:

$$diversity(\mathrm{L}(t), \mathrm{L}(t-\Delta t)) = \frac{\mid \mathrm{L}(t) \backslash \mathrm{L}(t-\Delta t) \mid}{\mathrm{N}} \tag{4.3}$$

dove Δt indica la distanza temporale tra le due liste TopN. La *novelty* invece misura quanto cambia una lista TopN nel tempo rispetto a tutte le raccomandazioni precedenti effettuate per l'utente: è definita quindi come:

$$novelty(\mathrm{L}(t)) = \frac{\mid \mathrm{L}(t) \backslash \{\cup_{0<x<t} \mathrm{L}(x)\}}{\mathrm{N}} \tag{4.4}$$

dove $\{\cup_{0<x<t} \mathrm{L}(x)\}$ indica l'unione di tutti gli *item* raccomandati all'utente dal tempo 0 (quando il sistema ha iniziato a funzionare) al tempo t (t escluso).

4.2 Perturbazione forzata

Per perturbazione forzata si intende una modifica artificiale dei dati in ingresso che non fa parte dell'evoluzione temporale del sistema. Una modifica forzata del modello può avvenire solo per sistemi collaborativi e non *Content-Based* (in quanto questi ultimi non basano le raccomandazioni sui *rating*) e può avere due scopi:

- Preservare la *privacy* dell'utente: dall'analisi dei *rating* presenti nella matrice URM è possibile identificare un utente: per questo motivo si introducono dei falsi voti senza che questi influenzino la lista di raccomandazione
- Attaccare il sistema: un aggressore potrebbe introdurre dei dati falsi nella matrice URM per fare in modo che il sistema di raccomandazione consigli un determinato *item*.

La perturbazione forzata del profilo utente invece rappresenta l'argomento di studio di questa tesi: modificando il profilo dell'utente abbiamo cercato di spiegare i film consigliati analizzando la diversità della lista di raccomandazione prima e dopo la modifica effettuata (capitolo 5), e sempre tramite la perturbazione del profilo abbiamo cercato di dare un valore di confidenza agli *item*, intesa come sicurezza da parte del sistema che il film consigliato possa piacere all'utente (capitolo 6). Nel primo caso la modifica del profilo utente consiste in una rimozione di un *item* tra quelli che l'utente ha visto, mentre nel secondo caso consiste nell'inserimento nel profilo di un film che gli è stato suggerito ma che ancora non ha visto.

Capitolo 5

Metodologia per le spiegazioni

In questo capitolo viene affrontato il problema delle spiegazioni: si cerca di spiegare i film consigliati all'utente tramite un approccio *Item-Based*, ossia cercando delle relazioni tra i film che ha visto e quelli che gli sono stati consigliati. Per questo scopo sono stati usati due metodi: il secondo altro non è che un'ottimizzazione del primo (sotto l'aspetto del tempo impiegato e del numero di confronti eseguiti) che, come mostrato in seguito, ha dato risultati nettamente migliori dal punto di vista delle prestazioni, penalizzando di poco l'accuratezza delle spiegazioni date. Il confronto tra i tempi di esecuzione e il numero di test necessari dei due metodi è descritto nella sezione 5.4.

5.1 Metodo normale

La prima prova che abbiamo fatto per spiegare un film i consigliato ad un utente u è stata quella di provare a rimuovere dal profilo dell'utente in questione un film a caso tra quelli visti (ponendo il suo voto pari a zero) e vedere cosa succede ad i se si effettua un'altra raccomandazione per quell'utente, ossia se è ancora consigliato o no: questo perché gli algoritmi di raccomandazione usano come input, oltre al modello, anche il profilo utente, ci si aspetta quindi che modificandolo si modifichi anche la raccomandazione effettuata. L'algoritmo prova così a rimuovere, uno alla volta, tutti i film visti dall'utente dal suo profilo : se l'*item* i durante una o più di queste rimozioni

dovesse sparire dalla nuova lista di raccomandazione Top5 creata a partire dal profilo modificato, si tiene conto del (o dei) film che sono "responsabili" del fatto che sia consigliato. Definendo N come il numero di film visti da un utente, in questa fase si eseguono quindi N prove, ossia si calcolano N liste di raccomandazione. Se invece il film non dovesse mai sparire dalle raccomandazioni si passa a rimuovere dal profilo tutte le possibili coppie di film visti dall'utente; se anche in questo modo non dovessimo riuscire a spiegare il film i, si passa a rimuovere dal profilo tre film alla volta, in tutte le possibili combinazioni. In questo caso, il numero delle possibili coppie o triple è dato da $\binom{N}{k}$ con k uguale a due o tre. E' stato deciso di provare a rimuovere uno (o due, o tre) alla volta tutti i film, e di non fermarsi appena i fosse stato spiegato, in quanto se ci fossimo fermati avremmo privilegiato nelle spiegazioni i film che nel dataset vengono per primi: abbiamo invece deciso di mostrare all'utente tutti i film che, se rimossi, fanno sì che un *item* non venga più consigliato.

La scelta della lunghezza dei profili mostrati nel capitolo 3.1 è stata fatta tenendo conto di questo test: il profilo A infatti non comprende utenti che hanno visto uno o due film in quanto non sarebbe stato possibile toglierne tre dal suo profilo e quindi i risultati non sarebbero stati confrontabili con quelli degli altri profili. L'altro vincolo, ossia il numero massimo di film visti dagli utenti posto pari a venti (profilo C), è dato dal tempo richiesto per i test: ad esempio, con 20 film visti, nella terza fase dell'algoritmo sono svolti $\binom{20}{3}$ test, ossia 1140; considerando anche solo gli utenti che hanno visto 21 film, questo limite massimo si sarebbe innalzato a 1330 test. Per gli stessi due motivi non è stata presa in considerazione l'ipotesi di togliere i film quattro alla volta dal profilo utente. In seguito è mostrato un esempio dell'output del programma per un utente di profilo A, usando il dataset di *MovieRec* (composto da film con titoli in italiano):

```
FILM VISTI:
La ragazza del lago
I  Tenenbaum
Next

FILM CONSIGLIATI:
Michael Clayton
Lo spaccacuori
```

```
Mr. Brooks
Prospettive di un delitto
Non è un paese per vecchi

SPIEGAZIONI:
Michael Clayton perché hai visto: La ragazza del lago
Lo spaccacuori perché hai visto: I  Tenenbaum
Mr. Brooks perché hai visto: Next
Prospettive di un delitto perché hai visto: Next
Non è un paese per vecchi perché hai visto: La ragazza del lago
```

Questo metodo risulta indipendente dal tipo di dataset e dall'algoritmo usati: è quindi stato applicato sia al dataset di *Netflix* in versione ridotta (con gli algoritmi Item-Item Cosine kNN, Item-Item Pearson, Asymmetric SVD e Pure SVD) sia a quello di *MovieRec* (con gli algoritmi Item-Item Cosine kNN, Pure SVD e LSA Cosine). Nelle figure 5.1, 5.2, 5.3 e 5.4 sono mostrati alcuni risultati: le figure presentano una colonna per ogni tipo di profilo utente (come definiti nel capitolo 3.1) con la percentuale di film spiegati rimuovendo un film dal profilo stesso (blu), rimuovendone due (azzurro), rimuovendone tre (giallo); in rosso è mostrata la percentuale di film che non è possibile spiegare in quanto necessitano più di tre rimozioni dal profilo utente.

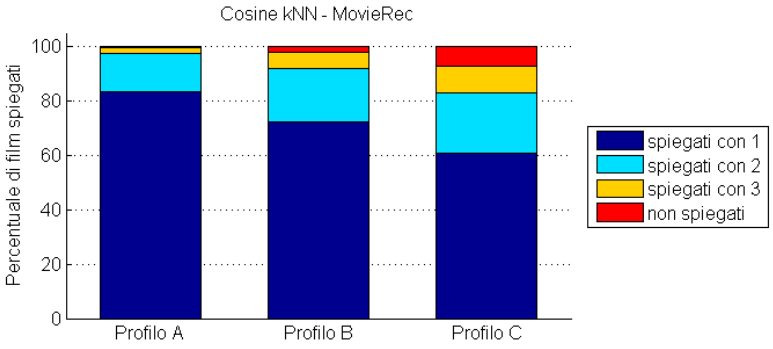


Figura 5.1: Percentuale di film spiegati, Item-Item Cosine kNN, MovieRec

In figura 5.1 si analizza l'algoritmo Item-Item Cosine kNN sul dataset di *MovieRec*: per gli utenti appartenenti al profilo B, ad esempio, oltre il

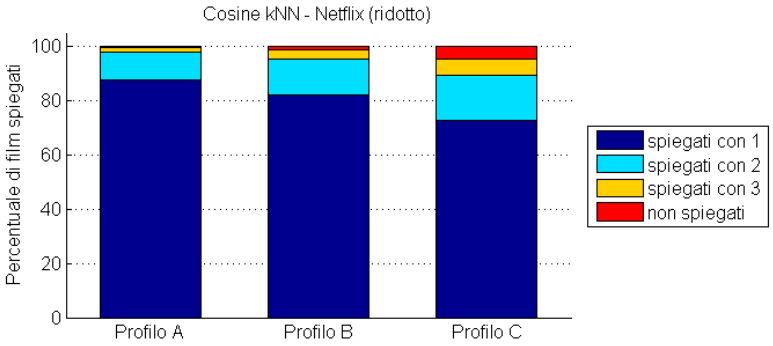


Figura 5.2: Percentuale di film spiegati, Item-Item Cosine kNN, Netflix (ridotto)

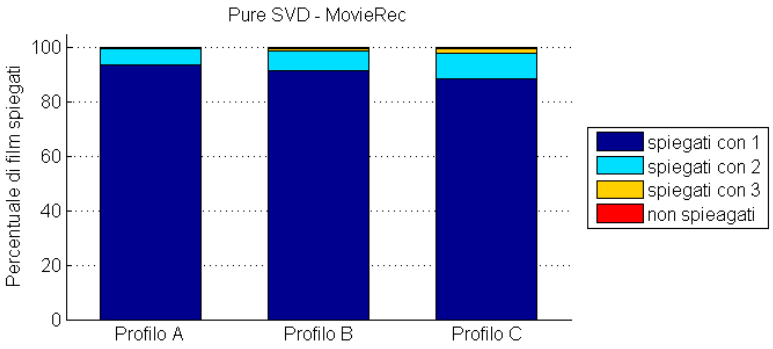


Figura 5.3: Percentuale di film spiegati, Pure SVD, MovieRec

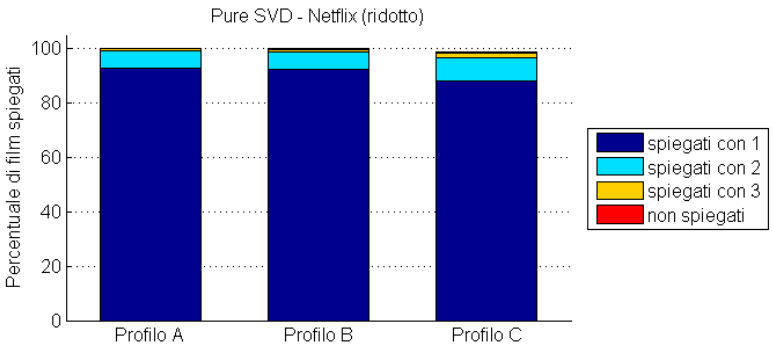


Figura 5.4: Percentuale di film spiegati, Pure SVD, Netflix (ridotto)

70% di film raccomandati può essere spiegato rimuovendo soltanto un film visto dal profilo, circa il 20% rimuovendone due, un altro 5% rimuovendone tre e la percentuale di film che non è possibile spiegare è quindi bassa, circa un 2% del totale dei film raccomandati. Dalle figure è immediato notare come cali la percentuale di film che è possibile spiegare rimuovendo solo un *item* dal profilo utente, aumentando il numero di film visti dagli utenti, ossia passando dagli utenti di profilo A, a quelli di profilo B e di profilo C: questo dato si spiega con il fatto che, essendo la raccomandazione basata sulla totalità dei film visti da un utente, è normale che per i profili A, ossia per gli utenti che hanno visto pochi film (da 3 a 5), una modifica del profilo relativa ad un film rappresenti una modifica consistente del profilo (che viene modificato fino al 33%), il che comporta con alta probabilità un cambiamento nella lista di raccomandazione. Al contrario, per gli utenti di profilo C (che hanno visto da 11 a 20 film), spesso rimuovere un solo film tra quelli visti comporta una modifica insignificante del profilo e questo spiega il perché di una percentuale bassa di film che è possibile spiegare con una sola rimozione.

Dai grafici mostrati si può vedere come la strada intrapresa sembri essere corretta per spiegare le raccomandazioni, sia per la bassissima percentuale di film che in questo modo non si riescono a spiegare, sia perché vengono soddisfatti alcuni degli scopi elencati da Tintarev (et al) [23] che dovrebbe avere un sistema di raccomandazione, tra cui:

- Fiducia: un sistema di spiegazione che ammette di non riuscire a spiegare un *item* suggerito risulta all'utente più sincero di un sistema che si limita a fare le raccomandazioni.

- Efficienza: un utente, in base alla spiegazione ricevuta, può decidere subito se vale la pena oppure no di vedere un certo film, in quanto collegato ad uno appartenente al suo profilo, di cui ha sicuramente le idee chiare su quanto gli sia piaciuto.

- Soddisfazione: le spiegazioni date in questo modo sono sicuramente facili da capire in quanto sono basate sui film da lui visti, non richiedono quindi un grande sforzo da parte dell'utente.

L'unico inconveniente emerso da questo tipo di test è il tempo richiesto eccessivo, non adatto ad un sistema che dovrebbe generare le raccomandazioni

(con le relative spiegazioni) in *Real-time*: è stato pensato quindi un metodo per ovviare a questo problema, mostrato nella sezione seguente.

5.2 Metodo ottimizzato

Partendo dai risultati precedenti, è stata creata una tabella contenente i film visti e quelli consigliati, e per ogni coppia l'indice di similitudine tra i due: è subito emerso che spesso il film che spiega uno consigliato è il più simile ad esso. E' quindi stata fatta una modifica all'algoritmo precedente: per ogni film consigliato i ad un utente u si cerca nel suo profilo il film visto m più simile ad i e si pone il suo *rating* pari a zero. Si ricalcola quindi una nuova lista di raccomandazione Top5 dando in input all'algoritmo il profilo modificato dell'utente u cercando in questa nuova lista il film i: se non è presente ci fermiamo dicendo che il film m spiega il film i, altrimenti andiamo a rimuovere dal profilo dell'utente un altro film, il più simile tra quelli rimasti (che chiameremo n), ricalcoliamo una nuova lista Top5 e cerchiamo nuovamente i in questa lista: se non è presente, m e n spiegano i. Se dovesse essere presente facciamo l'ultima modifica al profilo togliendo un altro film (o) e vedendo se con questa ulteriore modifica il film i è ancora presente: in caso negativo, possiamo dire all'utente che i tre film (m, n e o), spiegano il film i, altrimenti il film i è classificato come non spiegabile.

Con questo metodo ci si aspetta che i film non spiegati siano in numero maggiore rispetto a quelli non spiegabili con il primo metodo utilizzato, a vantaggio però di un tempo di esecuzione di molto ridotto: infatti con il primo metodo, prima di classificare come non spiegabile un film di un utente di profilo C (caso pessimo), erano necessari $\binom{20}{1}+\binom{20}{2}+\binom{20}{3}$ test, ossia 1350, ciascuno dei quali comportava il calcolo di una lista di raccomandazione, mentre con questo nuovo metodo al massimo si fanno solo 3 test (trascurando il tempo di calcolo della matrice di similitudine, in quanto in alcuni algoritmi si trova nel modello, in altri è ricavabile in un tempo trascurabile se diviso per la totalità degli utenti del dataset).

Nelle figure 5.5, 5.6, 5.7 e 5.8 sono mostrati i risultati ottenuti con il metodo ottimizzato relativi agli algoritmi Item-Item Cosine kNN e Pure SVD usando i dataset di *MovieRec* e *Netflix*. La figura 5.8 ad esempio è relativa all'algoritmo Pure SVD usato con il dataset di Netflix in versione ridotta: si può vedere come in questo caso le percentuali di film non spiegati

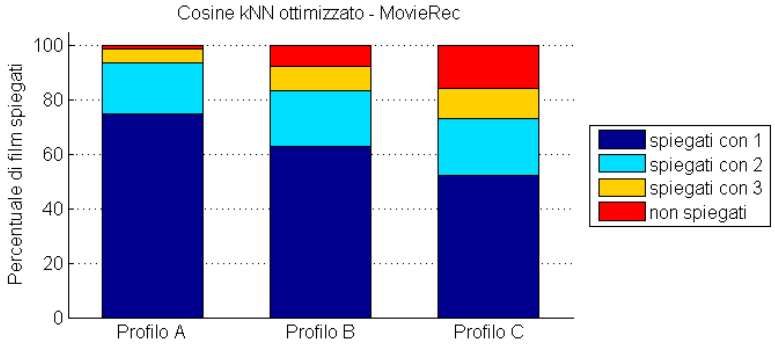


Figura 5.5: Percentuale di film spiegati, Item-Item Cosine kNN, MovieRec, metodo ottimizzato

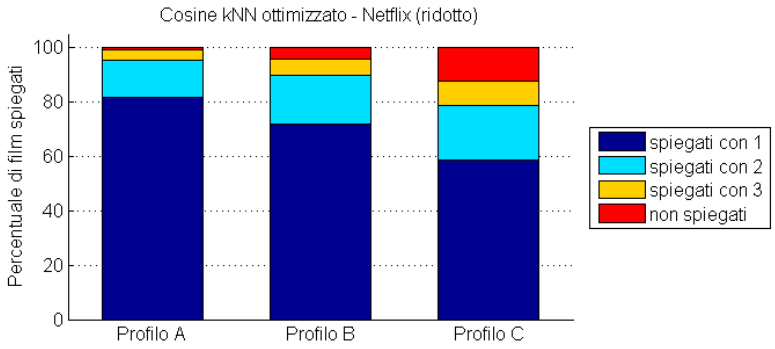


Figura 5.6: Percentuale di film spiegati, Item-Item Cosine kNN, Netflix (ridotto), metodo ottimizzato

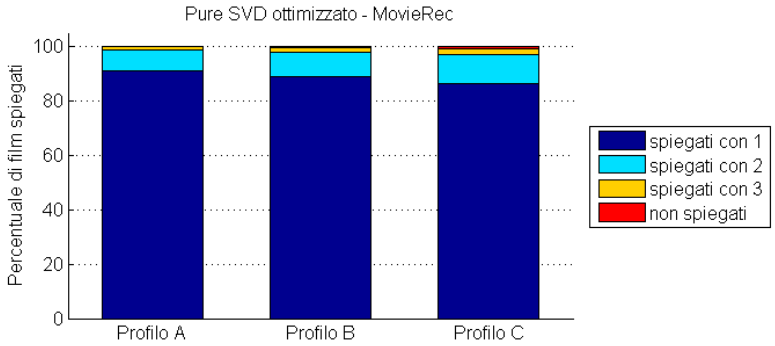


Figura 5.7: Percentuale di film spiegati, Pure SVD, MovieRec, metodo ottimizzato

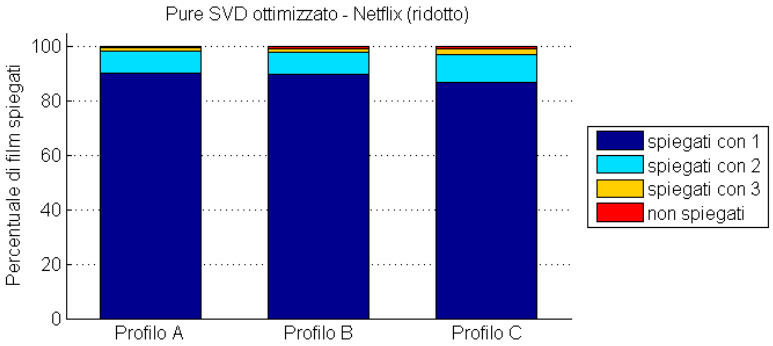


Figura 5.8: Percentuale di film spiegati, Pure SVD, Netflix (ridotto), metodo ottimizzato

non aumentino molto aumentando il numero di film visti dagli utenti, come invece avviene con l'altro algoritmo, in figura 5.6.

5.3 Risultati

Nelle tabelle 5.1 e 5.2 sono mostrate le spiegazioni date ad un utente di profilo B, rispettivamente relative alla lista Top5 creata con gli algoritmi Item-Item Cosine kNN e LSA Cosine. Le tabelle 5.3 e 5.4 invece mostrano le spiegazioni delle raccomandazioni effettuate con gli stessi due algoritmi, ma ad un utente di profilo C. Sulle righe di queste tabelle sono presenti i film (del dataset di MovieRec, con i titoli in italiano) visti dall'utente, mentre sulle colonne i 5 film che gli sono stati suggeriti dall'algoritmo di raccomandazione. Nella cella di intersezione tra una riga i ed una colonna j è presente una **N** se il film i spiega j con il metodo normale, una **O** se invece lo spiega con il metodo ottimizzato; se la cella non presenta alcun simbolo, il film i non spiega il film j. Analizzando ad esempio la tabella 5.1 si può notare che il film "Un'impresa da Dio" è spiegato con il metodo normale dai film "I racconti del brivido" e "Idiocracy", mentre per spiegare lo stesso film con il metodo ottimizzato sono necessari tre film: "SuXbad", "Molto incinta" e "Idiocracy". Dalla stessa tabella si può notare anche come non sia possibile spiegare il film "American Pie" con nessuno dei due metodi proposti.

Nella tabella 5.4 è invece possibile osservare come, usando l'algoritmo LSA Cosine per stilare una lista di raccomandazione per un utente che ha visto 11 film (ossia di profilo C), il film visto "L'alba dei morti viventi" sia predominante nelle spiegazioni dei film consigliati, spiegando da solo 4 film consigliati su 5 usando il metodo normale, e 3 usando quello ottimizzato (per spiegare "Lake Placid 2 Il terrore continua" infatti deve essere rimosso dal profilo dell'utente insieme al film "Il figlio di Chucky": questo ci fa capire che in realtà il film più simile a "L'alba dei morti viventi" sia appunto "Il figlio di Chucky").

Nelle tabelle 5.5 e 5.6 sono presenti i risultati riassuntivi delle percentuali di film che si possono spiegare con ciascun algoritmo usato, rispettivamente relative al dataset di *MovieRec* e a quello di *Netflix*. Ogni riga di queste tabelle rappresenta i risultati ottenuti con un diverso algoritmo, e la dicitura "ottimizzato" vicino al nome dell'algoritmo indica che i dati sono relativi al metodo ottimizzato (sezione 5.2). Per ogni algoritmo sono presenti i risultati relativi ai tre profili utente analizzati e per ciascuno di questi gruppi è visualizzata la percentuale di film che è possibile spiegare con una, due o tre rimozioni di film dal profilo. La colonna "> 3" indica la percentuale

Consigliati / **Visti**	American Pie	Un'impresa da Dio	Matrimonio alle Bahamas	Lo spaccacuori	Come tu mi vuoi
Prima ti sposo, poi ti rovino					
Una moglie bellissima					
Alvin Superstar			N - O		
SuXbad		O			
Molto incinta		O	O	N - O	N - O
I racconti del brivido		N			
Idiocracy		N - O			

Tabella 5.1: Esempio di spiegazioni dei film nella Top5: profilo B, Item-Item Cosine kNN

Consigliati / **Visti**	Princess	4 matrimoni e un funerale	Amori e sparatorie	Nel fantastico mondo di Oz	Lars e una ragazza tutta sua
Prima ti sposo, poi ti rovino					
Una moglie bellissima		N	N	N - O	
Alvin Superstar	N		N		
SuXbad	N				N - O
Molto incinta	N	N			
I racconti del brivido		N			
Idiocracy					

Tabella 5.2: Esempio di spiegazioni dei film nella Top5: profilo B, LSA Cosine

Visti \ **Consigliati**	Lo squalo	Stuart Little Un topolino in gamba	Rocky IV	La casa nera	Hulk
L'alba dei morti viventi					
Stuart Little 2		N - O			N - O
Il figlio di Chucky	N			N - O	
Jurassic Park III	N				
Edward mani di forbice					
Johnny English					
L'era glaciale					
Pattuglia di confine					
Spider-Man 2					
Carrie lo sguardo di satana	N				
Rocky III			N - O		

Tabella 5.3: Esempio di spiegazioni dei film nella Top5: profilo C, Item-Item Cosine kNN

Visti \ **Consigliati**	Rocky II	Anatomy	28 settimane dopo	Lake Placid 2 Il terrore continua	Wrong Turn 2 Senza via d'uscita
L'alba dei morti viventi		N - O	N - O	N - O	N - O
Stuart Little 2					
Il figlio di Chucky				O	
Jurassic Park III					
Edward mani di forbice					
Johnny English					
L'era glaciale					
Pattuglia di confine					
Spider-Man 2					
Carrie lo sguardo di satana					
Rocky III	N - O				

Tabella 5.4: Esempio di spiegazioni dei film nella Top5: profilo C, LSA Cosine

di film che non si è riusciti a spiegare neanche rimuovendo tre film dal profilo dell'utente. Dalla tabella 5.5 si può notare che per quanto riguarda la versione ottimizzata del metodo per spiegare i film eseguito sull'algoritmo LSA Cosine applicato al dataset di *MovieRec*, agli utenti di profilo A non è stato possibile spiegare lo 0,42% dei film che gli sono stati consigliati. In entrambe le tabelle è possibile notare come la percentuale di film spiegati con il metodo normale effettuando una solo rimozione dal profilo degli utenti sia solo di poco maggiore rispetto alla percentuale di film spiegati con il metodo ottimizzato: per quanto riguarda i risultati relativi al dataset di *MovieRec*, l'algoritmo in cui questa differenza è minima è Pure SVD, in cui lo scarto tra la percentuale di film spiegati con il primo metodo e quella di film spiegati con il metodo ottimizzato è sempre inferiore al 3% e inoltre si mantengono bassissime le percentuali (meno dell'1%) di film che non si riescono a spiegare con tre film, con entrambi i metodi. Per quanto riguarda il dataset di *Netflix* invece si sono ottenuti ottimi risultati con l'algoritmo Item-Item Pearson, con il quale le percentuali di film non spiegati si sono mantenute bassissime con entrambi i metodi. Il dato più significativo è la percentuale di film che non è possibile spiegare, che si mantiene sempre molto bassa tranne in alcuni casi, ossia quelli evidenziati nelle tabelle: per quanto riguarda il dataset di *MovieRec*, la versione ottimizzata del metodo applicato all'algoritmo Item-Item Cosine kNN non ha dato buoni risultati per gli utenti di profilo B e C, con un totale di film non spiegati rispettivamente pari al 7,56% e 15,6%, valori di molto superiori a quelli ottenuti con il metodo non ottimizzato. Anche per gli utenti di profilo C usando il metodo ottimizzato applicato all'algoritmo LSA Cosine i valori non sono buoni. In tabella 5.6 sono sempre evidenziati i valori di film non spiegati ritenuti troppo alti: come per l'altro dataset, questi valori sono associati al metodo ottimizzato e agli utenti di profilo B o C.

	Profilo A				Profilo B				Profilo C			
Spiegati con (#film):	**1**	**2**	**3**	**> 3**	**1**	**2**	**3**	**> 3**	**1**	**2**	**3**	**> 3**
Cosine kNN	83,34	14,37	2,09	0,2	72,38	19,41	6,02	2,19	60,75	22,13	10,02	7,1
Cosine kNN ottimizzato	74,92	18,74	5,27	1,07	62,91	20,61	8,92	7,56	52,4	20,77	11,23	15,6
Pure SVD	93,87	5,68	0,42	0,03	91,49	7,49	0,88	0,15	88,53	9,34	1,75	0,38
Pure SVD ottimizzato	91,27	7,71	0,94	0,08	88,96	9,2	1,47	0,37	86,4	10,62	2,25	0,73
LSA	94,42	5,12	0,44	0,02	87,74	10,39	1,52	0,34	75,21	18,34	4,74	1,71
LSA ottimizzato	85,25	11,87	2,46	0,42	72,64	18,11	6,07	3,18	55,31	22,82	10,91	10,96

Tabella 5.5: Percentuali di film spiegati, dataset di MovieRec

	Profilo A				Profilo B				Profilo C			
Spiegati con (#film):	**1**	**2**	**3**	**> 3**	**1**	**2**	**3**	**> 3**	**1**	**2**	**3**	**> 3**
Cosine kNN	87,74	10,02	1,97	0,26	82,12	13,27	3,36	1,24	72,66	16,95	5,93	4,47
Cosine kNN ottimizzato	81,69	13,60	3,89	0,82	71,99	17,74	6,22	4,05	58,63	19,99	9,23	12,15
Pure SVD	92,90	6,23	0,79	0,08	92,22	6,55	0,95	0,28	88,19	8,68	1,59	0,54
Pure SVD ottimizzato	90,29	8,07	1,44	0,20	89,82	8,06	1,52	0,60	87,01	9,91	2,12	0,96
Pearson	94,65	4,89	0,43	0,03	93,62	5,59	0,7	0,09	92,54	6,3	0,96	0,19
Pearson ottimizzato	93,23	6,1	0,62	0,05	90,19	8,04	1,44	0,33	87,14	9,29	2,39	1,18
Asymmetric SVD	87,46	9,9	2,28	0,35	80,03	13,91	4,11	1,96	72,4	17,21	5,99	4,4
Asymmetric SVD ottimizzato	78,89	14,1	5,47	1,54	69,92	16,69	7	6,39	61,5	18,76	8,41	11,33

Tabella 5.6: Percentuali di film spiegati, dataset di Netflix (ridotto)

5.4 Complessità dei due metodi a confronto

Dopo aver calcolato la percentuale di film spiegati grazie ai due metodi proposti, abbiamo deciso di valutare le prestazioni in termini di tempo medio e di numero medio di test effettuati per spiegare un item ad un utente; questi calcoli sono stati effettuati per ogni profilo utente di ogni algoritmo utilizzato.

Per quanto riguarda il numero medio di test (ossia di raccomandazioni) effettuati per spiegare un film, è necessaria una distinzione tra il metodo normale e quello ottimizzato. Per il metodo normale è stato calcolato in questo modo:

$$\#test = \frac{\#film\_racc \times \#rating + ns1 \times \binom{\#rating}{2} + ns2 \times \binom{\#rating}{3}}{\#utenti \times 5} \quad (5.1)$$

dove:

- $\#rating$ indica la lunghezza media del profilo degli utenti, inteso come numero medio di film visti.
- $ns1$ e $ns2$ indicano il numero di film totali del profilo utente preso in considerazione, che non è possibile spiegare rimuovendo rispettivamente uno o due film visti.
- $\#film\_racc$ indica il totale di film raccomandati agli utenti di quel profilo, calcolato come $\#utenti \times 5$

Il numero medio di test relativi al metodo ottimizzato è stato invece calcolato nel seguente modo:

$$\#test = \frac{\#film\_racc + ns1 + ns2}{\#utenti \times 5} \quad (5.2)$$

Per calcolare il tempo medio per spiegare un *item* è bastato dividere il tempo impiegato per spiegare tutti i film a tutti gli utenti di un profilo (A, B o C), per il numero degli utenti moltiplicato per 5, in quanto ad ogni utente è stata presentata una lista di raccomandazione top5:

$$tempo\_medio = \frac{tempo\_impiegato}{\#utenti \times 5} \quad (5.3)$$

Il *tempo_impiegato* è stato calcolato da un'apposita funzione Matlab in fase di esecuzione del nostro programma, ed è stato calcolato nello stesso modo sia per il metodo normale che per quello ottimizzato.

Nelle tabelle 5.7 e 5.8 sono mostrati i valori riassuntivi del numero di test effettuati per spiegare un *item* ad un utente. In queste tabelle è presente su ogni riga un algoritmo usato e la dicitura "ottimizzato" vicino al nome dell'algoritmo indica che i valori mostrati sono relativi al metodo ottimizzato. I dati come al solito sono divisi in base al profilo degli utenti: A, B o C. Ad esempio, dalla tabella 5.7 si può notare che per gli utenti di profilo C, usando l'algoritmo Item-Item Cosine kNN, è necessario effettuare in media 121,18 raccomandazioni per spiegare un film tra quelli consigliati se si adotta il metodo normale; se invece si cerca di spiegare lo stesso film con il metodo ottimizzato di raccomandazioni ne sono necessarie soltanto 1,74. Questo dato molto basso (1,74) lo si può facilmente spiegare rifacendosi alla tabella 5.5: per lo stesso gruppo di utenti usando lo stesso algoritmo (profilo C, Item-Item Cosine kNN), con solo 2 rimozioni (e quindi effettuando 2 liste di raccomandazione) si riescono a spiegare oltre il 72% di film consigliati. Concentrandosi sui soli metodi ottimizzati applicati ai tre profili C di tabella 5.7, si può notare come questi tre dati rispecchino i valori della tabella 5.5: il numero di test effettuati maggiore infatti è relativo all'algoritmo Item-Item Cosine kNN, mentre il valore minore è relativo all'algoritmo Pure SVD (soltanto 1,16 test per spiegare un *item*), così come la percentuale di film spiegati effettuando una rimozione dal profilo utente più bassa (52,4%) sia riferita l'algoritmo Item-Item Cosine kNN e la più alta (86,4%) sia riferita all'algoritmo Pure SVD.

Un comportamento analogo a quello appena descritto si può notare in tabella 5.8, che mostra sempre il numero di test medio effettuati relativi al dataset ridotto di *Netflix*.

Algoritmo	**Prof. A**	**Prof. B**	**Prof. C**
Cosine kNN	4,75	18,37	121,18
Cosine kNN ottimizzato	1,31	1,53	1,74
Pure SVD	4,13	10,21	34,09
Pure SVD ottimizzato	1,09	1,12	1,16
LSA	4,1	11,55	64,34
LSA ottimizzato	1,17	1,36	1,66

Tabella 5.7: Numero medio di test (raccomandazioni) per spiegare un film, dataset di MovieRec

Algoritmo	Prof. A	Prof. B	Prof. C
Cosine kNN	4,46	14,44	89,19
Cosine kNN ottimizzato	1,23	1,39	1,63
Pure SVD	4,12	10,01	34,85
Pure SVD ottimizzato	1,1	1,11	1,15
Pearson	4,02	9,44	26,78
Pearson ottimizzato	1,07	1,11	1,16
Asymmetric SVD	4,45	15,16	85,13
Asymmetric SVD ottimizzato	1,28	1,43	1,58

Tabella 5.8: Numero medio di test (raccomandazioni) per spiegare un film, dataset di Netflix (ridotto)

Nelle tabelle 5.9 e 5.10 invece è mostrato il tempo medio in millisecondi impiegato per spiegare un *item* consigliato, rispettivamente con il dataset di *MovieRec* e con quello di *Netflix* (versione ridotta). La struttura di queste tabelle è simile a quella delle tabelle con i valori dei test effettuati: ogni riga corrisponde ad un algoritmo, si fa distinzione tra metodo normale e metodo ottimizzato e i valori riassuntivi sono relativi ai tre profili utenti analizzati. Da queste tabelle si può notare che il tempo per effettuare una raccomandazione dipende molto dall'algoritmo utilizzato: se consideriamo ad esempio gli algoritmi Item-Item Cosine kNN ed Asymmetric SVD, usando il metodo normale (che non comporta il calcolo della matrice di similitudine), dalla tabella 5.8 si può notare che il numero medio di test effettuati per gli utenti di profilo C vale rispettivamente 89,19 e 85,13, ossia due valori molto simili; analizzando invece il tempo impiegato dagli stessi due algoritmi per gli stessi profili C di tabella 5.10, si nota che questi valori sono molto diversi: 595,06 millisecondi per l'algoritmo Item-Item Cosine kNN contro i 205,87 millisecondi impiegati dall'Asymmetric SVD. Sempre riferendosi alle tabelle 5.9 e 5.10 si può notare come i tempi si riducano passando dall'utilizzo del metodo normale a quello del metodo ottimizzato, soprattutto usando il dataset di *Netflix*: in alcuni casi infatti il valore si riduce quasi di un fattore venti (algoritmo Item-Item Cosine kNN, profilo C). La diversità tra i tempi di queste due tabelle si spiega con il fatto che effettuare una raccomandazione usando il dataset di *MovieRec* risulta molto più veloce rispetto alla stessa operazione effettuata con il dataset di *Netflix* in versione ridotta, in quanto

contiene un numero di film circa 7 volte inferiore.

Algoritmo	**Prof. A**	**Prof. B**	**Prof. C**
Cosine kNN	2,02	9,2	76,59
Cosine kNN ottimizzato	4,18	4,87	5,89
Pure SVD	1,49	3,12	12,55
Pure SVD ottimizzato	3,92	4,4	5,16
LSA	1,65	4,38	34,21
LSA ottimizzato	4,07	4,76	5,69

Tabella 5.9: Tempo medio per spiegare un film (in millisecondi), dataset di MovieRec

Algoritmo	**Prof. A**	**Prof. B**	**Prof. C**
Cosine kNN	14,51	66,4	595,06
Cosine kNN ottimizzato	23,19	28,1	31
Pure SVD	12,55	38,84	166,67
Pure SVD ottimizzato	24,66	25,08	25,91
Pearson	23,1	93,88	500,66
Pearson ottimizzato	32,71	43,62	53,98
Asymmetric SVD	7,79	32,44	205,87
Asymmetric SVD ottimizzato	16,6	19,61	20,67

Tabella 5.10: Tempo medio per spiegare un film (in millisecondi), dataset di Netflix (ridotto)

Capitolo 6

Metodologia per la confidenza

6.1 Introduzione

In questo capitolo vengono descritti due metodi implementati per calcolare la confidenza di un film raccomandato: per confidenza si intende un valore, espresso in punti percentuali, che esprime quanto siamo sicuri che l'*item* consigliato sia effettivamente d'interesse per l'utente.

Come illustrato nel capitolo 2.6, in letteratura sono presenti due metodi per il calcolo della confidenza, che sono:

- Confidenza calcolata come varianza dei voti ottenuti da un film [1].
- Confidenza come numero di votazioni ottenute da un film [20].

Questi due metodi associano un valore di confidenza per ogni film, mentre i nostri metodi associano un valore per ogni film a seconda dell'utente al quale il film è suggerito, quindi sono funzione sia dell'utente sia del film, e non solo del film. Proprio per questo motivo non è stato possibile un confronto tra i nostri metodi e quelli presenti in letteratura: il confronto è stato però eseguito tra questi due metodi tramite il calcolo del coefficiente di correlazione di Pearson che ha dato come risultato $\rho = -0,0672$, mostrando quindi una non correlazione tra i metodi. Questa non correlazione si può notare in figura 6.1, dove ogni asterisco rappresenta un film presente nel catalogo di *Netflix*, ed è caratterizzato da due valori: sulle ordinate è presente

il valore della varianza dei suoi voti, mentre sulle ascisse è presente il numero di *rating* che ha ottenuto (per facilità di lettura i valori delle ordinate sono stati invertiti in quanto un'alta confidenza è data da una bassa varianza). Se i due metodi fossero in qualche modo correlati, l'addensamento di asterischi seguirebbe l'andamento della bisettrice: dal grafico è possibile vedere che non è così.

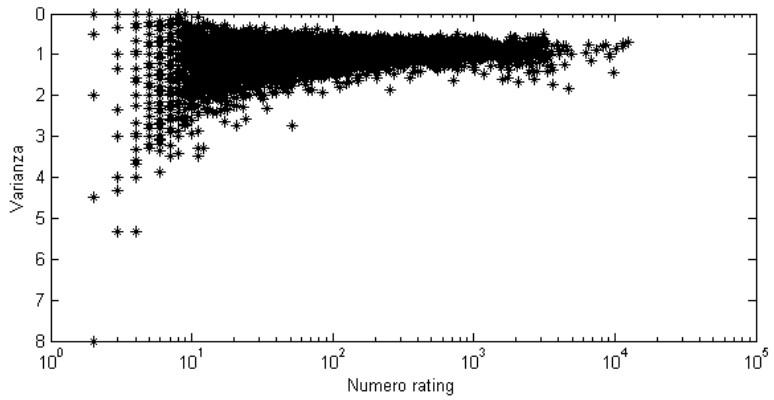


Figura 6.1: Confronto tra i due metodi trovati in letteratura per il calcolo della confidenza, dataset di Netflix (ridotto)

6.2 Confidenza: primo metodo e risultati

Un film suggerito si può dire avere alta confidenza se siamo sicuri che sia d'interesse per un utente, quindi se l'algoritmo glielo consiglia anche a fronte di stravolgimenti del suo profilo: è per questo motivo che abbiamo deciso di calcolare la confidenza del film raccomandato i come percentuale di film visti da un utente che vanno rimossi dal suo profilo affinchè l'algoritmo non gli consigli più di vedere i. Questo valore varia molto in base alla lunghezza della raccomandazione effettuata: se pensiamo ad un film presente in quinta posizione nella lista Top5, probabilmente basterà una leggera modifica del profilo affinchè questo film perda anche solo una posizione e non venga più consigliato: per questo motivo sono state effettuate diverse prove variando il parametro N, ossia la lunghezza della lista di raccomandazione; ovviamente ci si aspetta di ottenere risultati migliori per valori più alti di N. Per modificare il profilo dell'utente ci siamo basati sui risultati ottenuti dai due

metodi descritti nel capitolo precedente per la spiegazione delle raccomandazioni: abbiamo quindi deciso di togliere i film dal profilo utente iniziando con quello più simile al film di cui vogliamo calcolare la confidenza; se questa modifica non dovesse bastare per farlo sparire dalla TopN si procede quindi con il togliere il secondo film più simile, e così via finchè ci sono film da togliere nel profilo utente. In questo modo, una confidenza del 100% è stata associata ad un film consigliato i che continua ad essere consigliato nonostante siano stati rimossi tutti i film visti dall'utente: questo scenario è quasi impossibile e da ritenersi del tutto fortuito, in quanto può verificarsi solo se un algoritmo, a fronte di un profilo utente vuoto, consiglia casualmente un film che è lo stesso di cui si sta cercando di calcolare la confidenza.

Il comportamento degli algoritmi che ricevono in input un profilo vuoto, ossia di un utente che non ha visto alcun film, è riportato di seguito:

- Asymmetric SVD e Item-Item Pearson: i *rating* predetti per i film a catalogo sono tutti NaN, quindi la lista TopN è composta dai primi N film presenti nel catalogo: la confidenza del 100% si può avere solo per i primi N film.
- Item-Item Cosine kNN, Pure SVD e LSA Cosine: i *rating* predetti per i film a catalogo sono tutti 0, quindi la lista TopN è composta dai primi N film presenti nel catalogo: anche in questo caso la confidenza del 100% si può avere solo per i primi N film.

Questo metodo per calcolare la confidenza, che in seguito sarà chiamato Confidenza-1, è quindi dato da:

$$\textit{Confidenza-1}_{u,i} = \frac{\#film\_tolti_u}{\#rating_u} \times 100 \tag{6.1}$$

La Confidenza-1 è stata calcolata per ogni film presente nella lista Top5, calcolata con tutti gli algoritmi in nostro possesso, usando i dataset di *MovieRec* e di *Netflix* (versione ridotta). I valori relativi ai 5 film sono stati mantenuti separati in modo da poter calcolare poi il valor medio di Confidenza-1 relativo ad ognuna delle 5 posizioni della Top5: nelle figure 6.2 e 6.3 sono mostrati i risultati ottenuti usando il dataset di *MovieRec* e, rispettivamente, gli algoritmi Item-Item Cosine kNN e LSA Cosine, mentre nelle figure 6.4 e 6.5 i risultati ottenuti con gli algoritmi Asymmetric SVD e Item-Item Pearson applicati alla versione ridotta del dataset di *Netflix*. Sull'asse delle ascisse è possibile identificare i tre profili utente mentre sulle ordinate è

presente il valore, espresso in punti percentuali, di Confidenza-1, relativa ai 5 film della lista Top5: in blu è mostrata la confidenza media dei film che nelle liste di raccomandazioni di tutti gli utenti di un determinato gruppo occupano la prima posizione, mentre in azzurro, verde, arancione e marrone è visualizzato lo stesso valore, relativo rispettivamente ai film che nella lista di raccomandazione occupano la seconda, terza, quarta e quinta posizione.

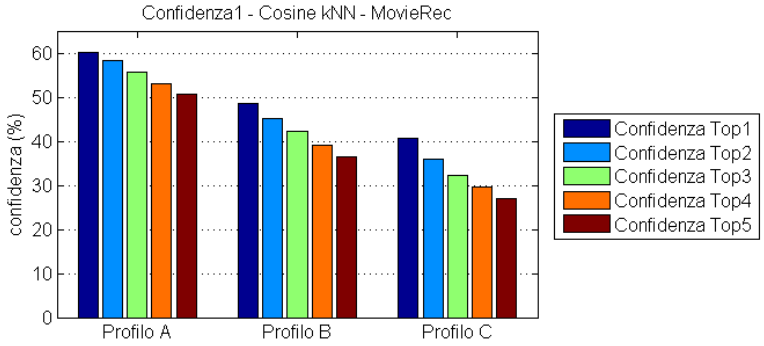


Figura 6.2: Confidenza-1 media per i 5 film della Top5, Cosine kNN, MovieRec

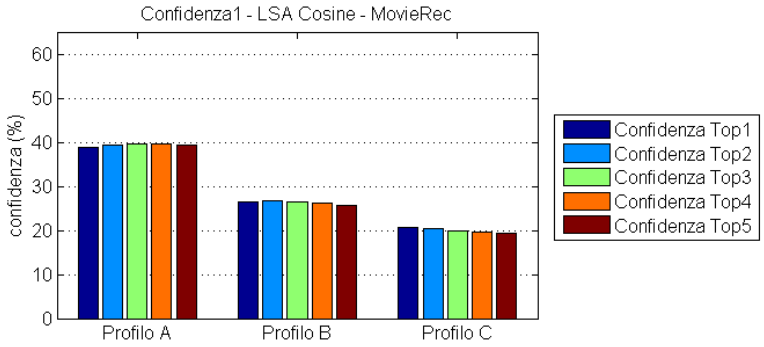


Figura 6.3: Confidenza-1 media per i 5 film della Top5, LSA Cosine, MovieRec

In figura 6.2, ad esempio, per gli utenti di profilo A si può notare che mediamente il primo film consigliato, ossia quello che secondo l'algoritmo di raccomandazione dovrebbe interessare maggiormente all'utente, ha un valore ci Confidenza-1 pari a oltre il 60%: questo dato significa che per far sì che questo film (ossia quello che occupa la prima posizione nella lista Top5) non venga più suggerito all'utente, dal suo profilo devono essere rimossi quasi i $\frac{2}{3}$ dei film che ha visto, partendo ovviamente dai più simili (basandosi

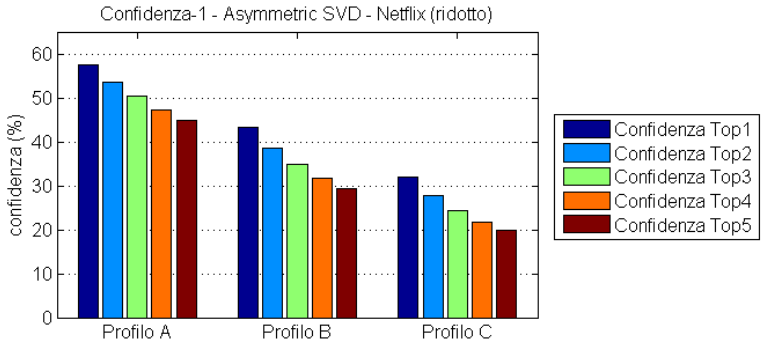


Figura 6.4: Confidenza-1 media per i 5 film della Top5, Asymmetric SVD, Netflix (ridotto)

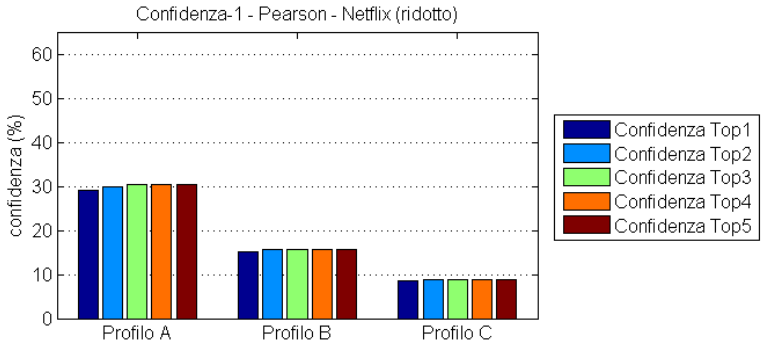


Figura 6.5: Confidenza-1 media per i 5 film della Top5, Pearson, Netflix (ridotto)

sulla matrice di similitudine) a quello che gli è stato suggerito. Naturalmente ci si aspetta (come si può notare dalle figure 6.2 e 6.4) che i valori di Confidenza-1 diminuiscano all'aumentare delle posizione occupata nella Top5 dai film consigliati: le figure 6.3 e 6.5 sono state inserite per mostrare il comportamento anomalo degli algoritmi LSA Cosine e Item-Item Pearson: è possibile notare come il valore medio di Confidenza-1 non sia monotono decrescente considerando i valori dei 5 film: Ad esempio in figura 6.5 si può notare come i valori di Confidenza-1 relativi agli utenti di profilo A, oltre ad essere molto più bassi degli stessi valori ottenuti con altri algoritmi, aumentino passando dai film che occupano la prima posizione a quelli che occupano l'ultima (quinta). Nelle figure 6.6, 6.7, 6.8 e 6.9 sono mostrati gli intervalli di confidenza per quanto riguarda i valori di Confidenza-1 mostrati nei grafici precedenti, senza fare più distinzione tra diversi profili utenti per una maggiore semplicità di lettura dei risultati.

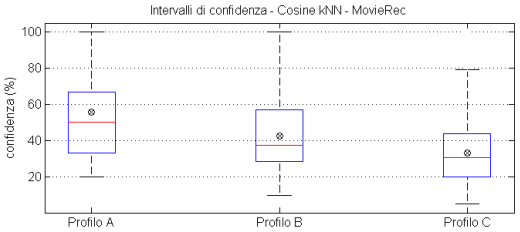


Figura 6.6: Intervalli di confidenza della Confidenza-1, Cosine kNN, MovieRec

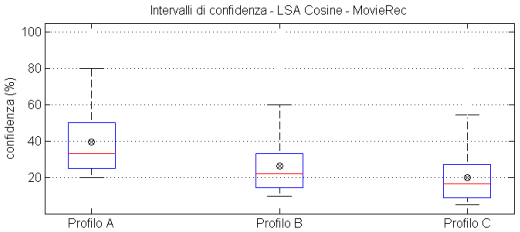


Figura 6.7: Intervalli di confidenza della Confidenza-1, LSA Cosine, MovieRec

In questi grafici per ogni profilo utente è disegnato un boxplot che riassume diversi valori: la linea interna al "contenitore" indica la mediana, i lati inferiore e superiore indicano rispettivamente il 25° e il 75° percentile, mentre la linea inferiore e superiore indicano il valore minimo e massimo.

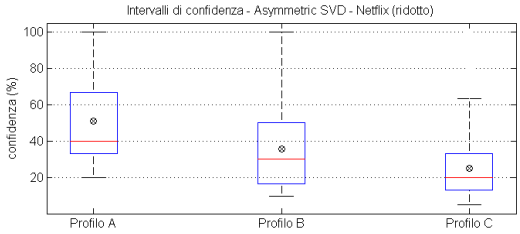


Figura 6.8: Intervalli di confidenza della Confidenza-1, Asymmetric SVD, Netflix (ridotto)

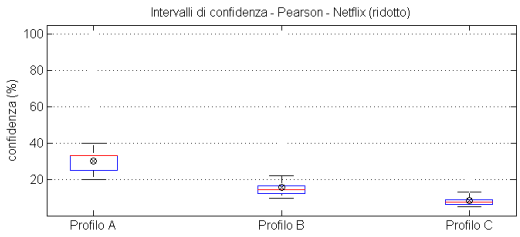


Figura 6.9: Intervalli di confidenza della Confidenza-1, Pearson, Netflix (ridotto)

Inoltre il cerchietto con una croce all'interno indica la media dei valori. I valori riassuntivi relativi a tutti gli algoritmi usati sono mostrati nelle figure 6.1 (per quanto riguarda il dataset di *MovieRec*) e 6.2 (per quanto riguarda il dataset di Netflix in versione ridotta).

Dopo aver calcolato la Confidenza-1 su una lista Top5, è stata calcolata anche per le liste Top10, Top15 e Top20: nelle figure 6.10 e 6.11 sono mostrati i valori medi di Confidenza-1 ottenuti senza fare distinzione tra diversi profili utente; sull'asse delle ascisse sono presenti i diversi valori della lunghezza della lista di raccomandazione: come ci aspettavamo, all'aumentare del parametro N aumenta anche la Confidenza-1 media dei film suggeriti. Per calcolare questi valori è stata apportata una piccola modifica al metodo usato per calcolare la Confidenza-1: inizialmente questo valore era rappresentato dalla percentuale di film da rimuovere sul totale di quelli visti da un utente affinchè l'*item* consigliato scendesse di posizione nella lista Top5 fino ad arrivare almeno alla sesta posizione; avendo ora allungato la lista di raccomandazione fino ad N film (con N uguale a 10, 15 o 20), anche il film da spiegare non deve essere presente nelle prime N posizioni. Questa modifica è stata necessaria in quanto altrimenti tutti i film presenti nella lista

	Profilo A				Profilo B				Profilo C			
Algoritmo	μ	q_{25}	q_{50}	q_{75}	μ	q_{25}	q_{50}	q_{75}	μ	q_{25}	q_{50}	q_{75}
Cosine kNN	55,77	33,33	50	66,67	42,49	28,57	37,5	57,14	33,19	20	33,77	43,75
Pure SVD	37,46	25	33,33	40	21,5	12,5	16,67	28,57	13,24	7,69	10,53	16,67
LSA Cosine	39,45	25	33,33	50	26,34	14,29	22,22	33,33	20,11	9,09	16,67	27,27

Tabella 6.1: Intervalli di confidenza della Confidenza-1, dataset di MovieRec

	Profilo A				Profilo B				Profilo C			
Algoritmo	μ	q_{25}	q_{50}	q_{75}	μ	q_{25}	q_{50}	q_{75}	μ	q_{25}	q_{50}	q_{75}
Cosine kNN	45,46	25	33,33	66,67	30,39	16,67	25	37,5	25,65	12,5	20	33,3
Pure SVD	37,09	25	33,33	40	20,36	12,5	16,67	50	12,48	7,14	9,09	15,79
Pearson	30,13	25	33,33	33,33	15,64	12,5	14,29	16,67	8,72	6,25	7,69	9,09
Asymmetric SVD	50,89	33,33	40	66,67	35,68	16,67	30	50	25,19	13,33	20	33,33

Tabella 6.2: Intervalli di confidenza della Confidenza-1, dataset di Netflix (ridotto)

di raccomandazione tra la sesta e la ventesima posizione avrebbero avuto Confidenza-1 pari a zero, essendo già al di fuori della Top5. Si può notare come con l'algoritmo Item-Item Person i valori della media siano pressochè inalterati cambiando il valore di N: questo dato può essere interpretato come sicurezza da parte dell'algoritmo della lista di raccomandazione presentata in base al profilo: se in effetti quest'ultimo viene modificato, i film suggeriti, qualora dovessero perdere posizioni nella lista TopN, ne perdono molte: bisogna però notare come questo livello di Confidenza-1 sia molto basso, soprattutto se messo in relazione a quello ottenuto dagli altri algoritmi. Alti valori di Confidenza-1 sono ottenuti invece da Asymmetric SVD per quanto riguarda il dataset di *Netflix* (figura 6.11) e da Item-Item Cosine kNN per quanto riguarda il dataset di *MovieRec* (figura 6.10).

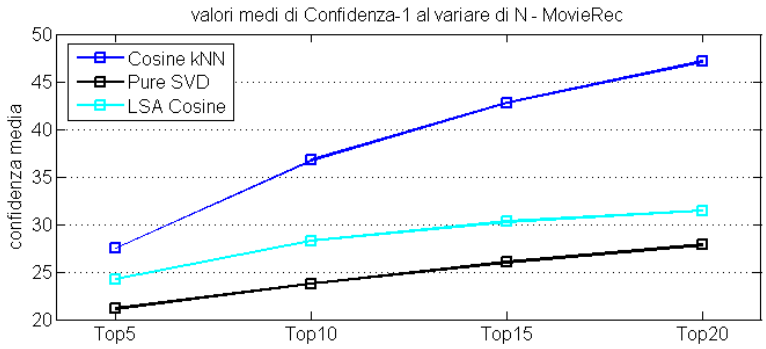


Figura 6.10: Valori medi di Confidenza-1 al variare del parametro N, MovieRec

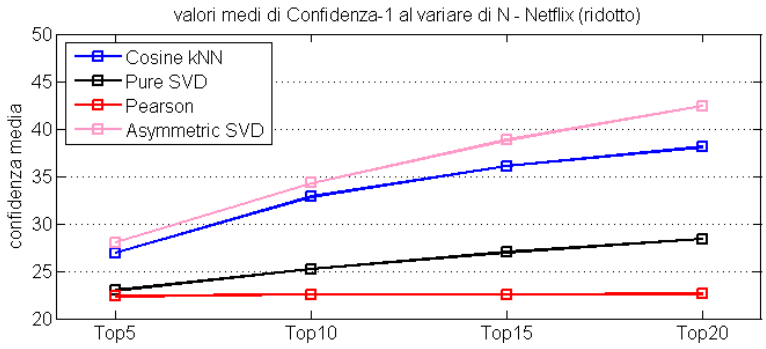


Figura 6.11: Valori medi di Confidenza-1 al variare del parametro N, Netflix (ridotto)

6.3 Confidenza: secondo metodo e risultati

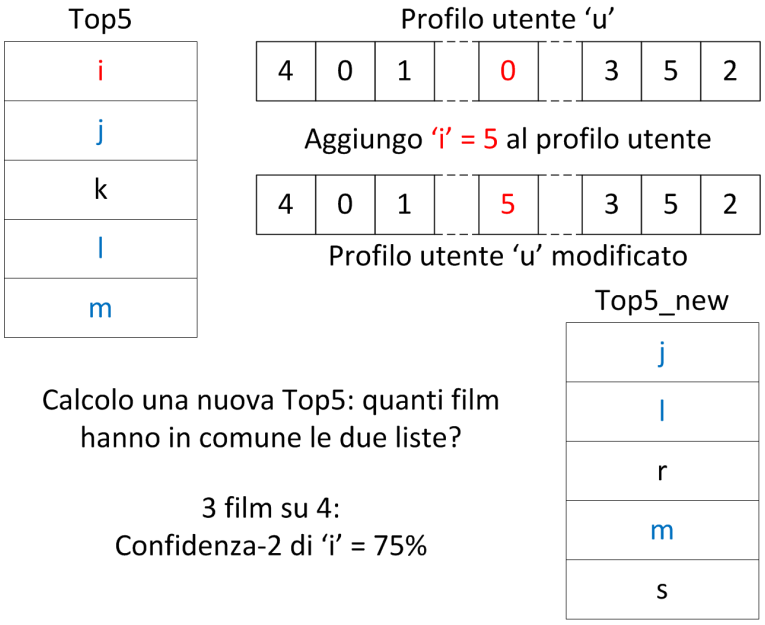


Figura 6.12: Schema del metodo adottato per il calcolo della Confidenza-2

Un secondo metodo per calcolare la confidenza relativa ai film raccomandati, chiamato Confidenza-2, è il seguente (figura 6.12): per calcolare la confidenza di i, aggiungo i al profilo utente con un voto pari a 5 per i dataset esplicito e pari a 1 per quello implicito, quindi vado a consigliare una nuova lista Top5 all'utente in base al profilo modificato: il grado di confidenza sarà quindi dato dal numero di film presenti in entrambe le liste (ossia quella ottenuta prima dell'aggiunta di i al profilo e quella ottenuta dopo). Per come è strutturato questo metodo, la confidenza massima si avrà quando le liste avranno in comune 4 elementi (5 è impossibile in quanto un algoritmo, giustamente, non consiglia ad un utente un film già presente nel suo profilo, i in questo caso). Abbiamo così deciso di associare ad i un valore pari al 100% qualora le due liste abbiano in comune 4 elementi, 75% se ne hanno in comune 3, 50% se ne hanno in comune 2, 25% se ne hanno in comune 1 e 0% se non ne hanno in comune.

$$\textit{Confidenza-2}_{u,i} = \frac{|\, L_1 \cap L_2 \,|}{4} \times 100 \tag{6.2}$$

con L_1 e L_2 rispettivamente liste Top5 calcolate prima e dopo l'inserimento di i nel profilo di u.

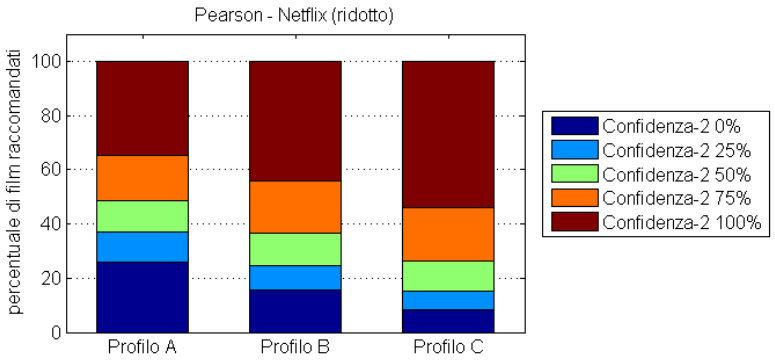


Figura 6.13: Percentuale di film della lista Top5 con un dato valore di Confidenza-2, Pearson, Netflix (ridotto)

Nella figura 6.13 è possibile notare i valori di Confidenza-2 relativi all'algoritmo Item-Item Pearson applicato al dataset ridotto di *Netflix*: sono presenti tre colonne, una per ogni profilo utente, ogni colonna è composta da diversi colori che indicano la percentuale di film presenti nella lista Top5 che hanno un determinato valore di Confidenza-2. Ad esempio, analizzando la figura 6.13 si può vedere che per quanto riguarda gli utenti di profilo A, circa il 25% dei film raccomandati (fascia blu) ha una Confidenza-2 dello 0%, che significa che se aggiunti al profilo utente con voto pari a 5, questi film sconvolgono completamente la lista di raccomandazione facendo sparire gli altri 4 (ossia tutti quelli della prima Top5 escluso il film di cui si sta calcolando la Confidenza-2) dalla nuova lista Top5. In azzurro, verde e arancione sono invece rappresentate le percentuali di film che hanno rispettivamente un valore di Confidenza-2 del 25%, 50% e 75%. Infine, in marrone, è evidenziata la percentuale di film con altissima Confidenza-2 (100%): questo valore ci mostra che circa il 35% dei film raccomandati a questi utenti, se visti e votati con voto pari a 5, non cambiano per niente una nuova lista che verrebbe creata dopo questa aggiunta al profilo utente.

Nella figura 6.14 invece si possono vedere i risultati del calcolo della Confidenza-2 applicati al dataset di *MovieRec* usando l'algoritmo Item-Item Cosine kNN: si può notare che hanno confidenza massima (100%) rispettivamente il 20%, 35% e 70% dei film visti dagli utenti di profilo A, B e C, mentre un basso livello di Confidenza-2 (fino al 50%) è ottenuto da una

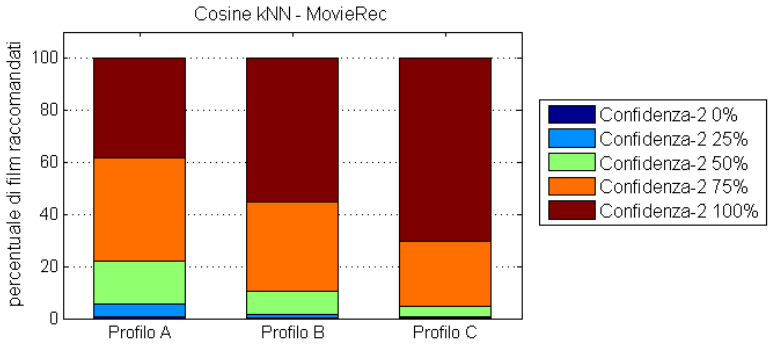


Figura 6.14: Percentuale di film della lista Top5 con un dato valore di Confidenza-2, Asymmetric SVD, Netflix (ridotto)

piccolissima percentuale di film della Top5, soprattutto per i profili degli utenti che hanno visto da 6 a 20 film (profili B e C), dove questo valore è ottenuto da meno del 10% del totale dei film raccomandati.

Capitolo 7

Confidenza e qualità delle raccomandazioni

In questo capitolo vengono prima introdotte le metriche usate per testare la bontà dei metodi usati per il calcolo della confidenza e infine, i risultati ottenuti sui dataset di *MovieRec* e *Netflix*, sia ridotto che completo.

7.1 K-fold cross validation

La K-fold cross validation è una tecnica usata per valutare come i risultati di un'analisi statistica possano generalizzare su dataset indipendenti [13]. Si inizia con il dividere il dataset (nel nostro caso rappresentato dalla matrice URM) orizzontalmente in K parti aventi la stessa cardinalità: una parte viene chiamata *test-set*, mentre le restanti k-1 vengono chiamate *train-set*. Il procedimento viene ripetuto K volte per far sì che ad ogni iterazione ci sia un *test-set* diverso e allo stesso tempo un *train-set* formato da k-1 fold presi in tutti i K modi possibili. Il *train-set* viene utilizzato per generare il modello che andrà fornito all'algoritmo di raccomandazione, mentre il *test-set* viene usato per testare la metrica scelta. Alla fine del test si potrà fare una combinazione dei K valori ottenuti: nel nostro caso si è deciso di fare una media e di usare un K pari a 10.

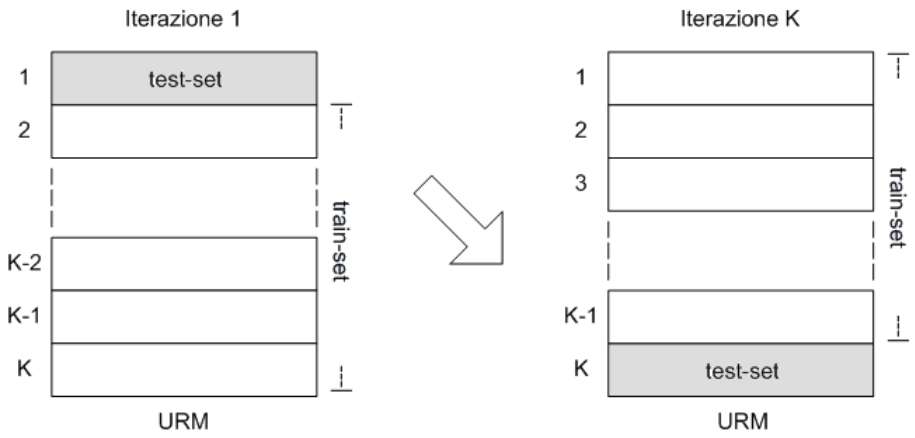


Figura 7.1: Esempio di K-fold cross validation

7.2 Recall

Ogni *item* consigliato ad un utente può essere rilevante oppure no: è rilevante se effettivamente è un *item* a cui, una volta visto, l'utente da un voto alto, al contrario non è rilevante se l'utente dopo averlo visto gli da un voto basso. I film possono quindi essere suddivisi in quattro categorie:

- **Veri Positivi (VP):** film consigliati che sono effettivamente d'interesse per l'utente.
- **Falsi Positivi (FP):** film consigliati che non sono d'interesse per l'utente.
- **Veri Negativi (VN):** film che il sistema non consiglia e che non sono d'interesse per l'utente.
- **Falsi Negativi (FN):** film che il sistema non consiglia ma che sono d'interesse per l'utente.

Una volta effettuata questa distinzione, è possibile calcolare la *Recall* [4]:

$$Recall = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{7.1}$$

La *Recall* rappresenta quindi la percentuale di film consigliati all'utente che sono di suo interesse, sulla totalità di film che sono di suo interesse, ed è la metrica più accurata per testare la bontà di un algoritmo di raccomandazione. Per il calcolo della *Recall* è stata usata la tecnica del *Leave one out*:

si considera un profilo utente, ossia i voti che l'utente ha espresso sui film che ha visto, si cerca un voto alto (nel nostro caso sono stati considerati alti i voti uguali a 5 presenti nel dataset esplicito di Netflix e quelli uguali ad uno nel dataset implicito di MovieRec) e si pone quel voto a zero. Si calcola quindi la lista di raccomandazione per quell'utente e se il film il cui voto è stato posto a zero (ossia il film che per l'algoritmo non è stato visto dall'utente) compare nella TopN, allora può essere considerato come vero positivo, altrimenti è un falso negativo. I risultati di questo test dipendono molto dal valore di N, ossia dalla lunghezza della lista di raccomandazione: nel nostro caso N è uguale a 5, ossia è stato mantenuto uguale al valore usato nei test fin qui svolti.

7.3 Fall-out

La *Fall-out*, contrariamente alla *Recall*, rappresenta la percentuale di falsi positivi per un utente sul totale di film che gli sono raccomandati [4]:

$$\textit{Fall-out} = \frac{\text{FP}}{\text{FP} + \text{VN}} \tag{7.2}$$

Anche in questo caso per calcolare i falsi positivi è stata usata la tecnica della *Leave one out*: questa volta però viene rimosso dal profilo dell'utente un film che ha visto e che non gli è piaciuto, ossia a cui l'utente ha dato un voto pari ad uno: se questo film viene consigliato dall'algoritmo di raccomandazione, ossia se compare nella TopN, è da considerarsi un falso positivo, altrimenti è un vero negativo. La *Fall-out* può essere calcolata solo su dataset espliciti, in quanto quelli impliciti ci danno informazioni soltanto sui film ritenuti interessanti per un certo utente, e non sui film che non gli sono piaciuti: per questo test è stato usato solo il dataset di Netflix, e come sempre il valore usato di N è pari a 5.

7.4 Metodologia per il calcolo di Recall e Fall-out

In questo capitolo viene descritto in che modo sono state calcolate la *Recall* e la *Fall-out* applicate alle liste di raccomandazioni ottenute tramite l'utilizzo dei diversi metodi per il calcolo della confidenza, mostrati nei capitoli precedenti:

- Primo metodo (Confidenza-1): dal profilo dell'utente si toglie un film che gli è piaciuto se si deve calcolare la *Recall* o che non gli è piaciuto se si deve calcolare la *Fall-out*, ponendo il suo voto pari a zero; viene quindi calcolata una lista di raccomandazione **Top20**: di questi 20 film si calcola il valore di confidenza come indicato nel capitolo 5.2, quindi si ordina la lista in base a questo valore e si tengono i primi 5 film, in modo da avere come sempre una lista Top5, usata per cercare la presenza del film tolto precedentemente.

- Secondo metodo (Confidenza-2): il procedimento è lo stesso del metodo precedente, la lista Top20 viene però ordinata in base al secondo metodo usato per calcolare la confidenza, quello mostrato nel capitolo 5.3, quindi vengono tenuti i 5 film con confidenza più alta.

- Confidenza come numero di *rating* ottenuti (SoA-1): in questo caso viene calcolato il numero di *rating* ottenuti da ogni film, quindi convertito in un valore espresso in percentuale relativa al valore massimo trovato e viene cercato il film tolto dal profilo dell'utente tra i 5 film con valore di confidenza più alto tra i primi 20 della lista di raccomandazione.

- Confidenza come varianza (SoA-2): in questo caso viene calcolata la varianza dei *rating* per ogni singolo film: questi valori vengono poi espressi in percentuale relativa al valore più basso trovato, fissato come 100%, e vengono tenuti i 5 valori più alti tra i primi 20: tra questi valori si cerca il film rimosso dal profilo utente per il calcolo di *Recall* o *Fall-out*.

Per il calcolo della *Recall* sono stati usati tre dataset: quello di MovieRec e due di Netflix: la versione ridotta e quella completa. Inoltre con il dataset di MovieRec non è stato possibile calcolare la confidenza basata sulla varianza dei *rating*, essendo impliciti e quindi tutti pari a uno. Per il calcolo della *Fall-out*, come detto in precedenza, è necessario disporre di *rating* espliciti, quindi sono state usate soltanto le due versioni del dataset di Netflix.

7.5 Risultati MovieRec

Il calcolo della *Recall* con il dataset di *MovieRec* è stato effettuato usando i tre algoritmi a nostra disposizione e testando il 40% dei *rating* pari a

5 presenti in ognuno dei 10 fold usati come test-set. I risultati mostrati in figura 7.2 e 7.3 sono stati ottenuti testando rispettivamente tutti i film e solo quelli non popolari, con gli utenti di profilo C: sulle ascisse sono presenti i tre algoritmi usati e sulle ordinate i relativi valori (espressi in punti percentuali) di *Recall*: i 4 diversi simboli identificano il modo in cui la *Recall* è stata calcolata:

- $*$: calcolata con il metodo standard.
- $+$: calcolata ordinando la lista Top20 in base al valore di Confidenza-1 e tenendo quindi i primi 5 valori ottenuti.
- o: calcolata ordinando la lista Top20 in base al valore di Confidenza-2 e tenendo quindi i primi 5 valori ottenuti.
- $\times$: calcolata ordinando la lista Top20 in base al numero di *rating* ottenuti dai film e tenendo quindi i primi 5 valori ottenuti.

Ad esempio in figura 7.2 per quanto riguarda l'algoritmo Item-Item Cosine kNN, si può notare come il metodo legato al calcolo della Confidenza-2 dia il risultato migliore, con una valore di *Recall* che vale circa 21%, seguito in ordine dal metodo standard (17,5%), da quello legato al numero di *rating* ottenuti (12,5%) e infine dal metodo legato al calcolo della Confidenza-1 (circa 4%). Confrontando i tre diversi algoritmi si può notare che il valore di *Recall* più alto sia ottenuto sempre tramite il calcolo della Confidenza-2 mentre il valore più basso tramite il calcolo della Confidenza-1; solo per l'algoritmo Pure SVD questa caratteristica non si nota subito in quanto ci sono due simboli sovrapposi: servendoci dei dati analitici si può però confermare quanto detto in precedenza, ossia che il metodo che usa i valori di Confidenza-2 restituisce un valore di *Recall* più alto con ogni algoritmo. Questa caratteristica si nota meglio in figura 7.3, dove il divario tra il valore ottenuto tramite il calcolo di Confidenza-2 e gli altri metodi è nettamente superiore per quanto riguarda l'algoritmo Pure SVD e pressochè invariato per gli altri 2 algoritmi: l'unico metodo penalizzato dalla scelta dei film non popolari come soggetti dei test è ovviamente quello basato sul numero di voti, che per sua natura tende a favorire i film popolari, com'è possibile notare confrontando le due figure.

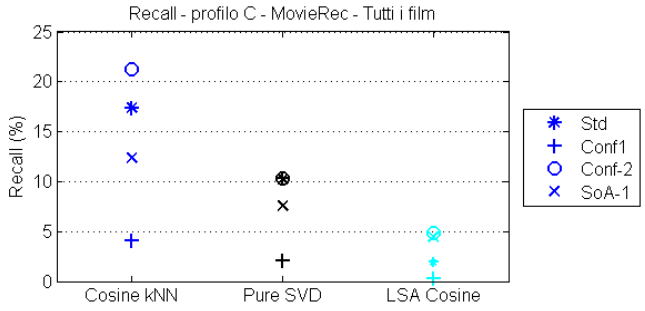


Figura 7.2: Recall calcolata con i diversi metodi di confidenza, dataset di MovieRec, tutti i film

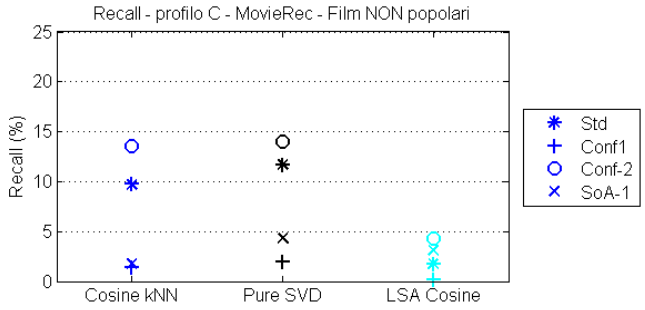


Figura 7.3: Recall calcolata con i diversi metodi di confidenza, dataset di MovieRec, film non popolari

7.6 Risultati Netflix (versione ridotta)

Con il dataset di *Netflix* in versione ridotta, trattandosi di un dataset esplicito, è stato possibile calcolare sia *Recall* che *Fall-out*: i risultati sono mostrati nelle figure 7.4, 7.5, 7.6 e 7.7, dove in ogni figura sono riassunti i risultati relativi a due algoritmi, usando nei test tutti i film (figure 7.4 e 7.6) o solo quelli non popolari (figure 7.5 e 7.7), per i soli utenti di profilo B. Ogni simbolo rappresenta un valore di *Recall* (espresso sulle ordinate) e di *Fall-out* (espresso sulle ascisse), relativo ad uno dei 5 metodi utilizzati, che sono gli stessi usati con il dataset di *MovieRec* più un quinto, espresso dal simbolo $\diamond$, che indica il valore calcolato tramite il metodo della letteratura che associa un valore di confidenza più alto ai film la cui varianza dei *rating* è bassa. In figura 7.6 e 7.7 sono mostrati i risultati relativi agli algoritmi Asymmetric SVD e Pure SVD (rispettivamente in rosa e in nero): per quanto riguarda Asymmetric SVD si può notare che, indipendentemente dal metodo usato, i valori di *Fall-out* sono molto bassi (di molto inferiori all'1%). Ricordiamo che i migliori risultati sono dati da un valore basso di *Fall-out* e alto di *Recall*: siccome la *Fall-out* per questo algoritmo varia di molto poco a seconda del metodo utilizzato, possiamo confrontare i diversi metodi basandoci solo sulla *Recall*, ossia possiamo affermare che il metodo che da risultati migliori sia (seppur di poco) quello basato sul calcolo della confidenza intesa come numero di *rating* ottenuti (SoA-1); come è facile aspettarsi il risultato cambia in figura 7.7, dove i film usati per i test sono soltanto quelli non popolari: la *Recall* calcolata tramite Confidenza-2 risulta essere la maggiore, mentre il risultato ottenuto tramite SoA-1 diminuisce notevolmente, passando da 4% a circa 2% .

7.7 Risultati Netflix (versione completa)

I risultati fin qui ottenuti sono stati calcolati su una versione ridotta del dataset di *Netflix* in quanto il calcolo di *Recall* e *Fall-out* sulla versione completa tramite *k-fold cross validation* avrebbe richiesto troppo tempo. Abbiamo comunque ritenuto necessario calcolare *Recall* e *Fall-out* anche sulla versione originale del dataset, in quanto le versioni ridotte sono ottenute scegliendo solo alcuni utenti e alcuni film del catalogo, quindi potrebbero dare risultati un po' differenti rispetto a quelli che si potrebbero ottenere con il dataset completo. Per effettuare i test sulla versione completa quindi è stato adot-

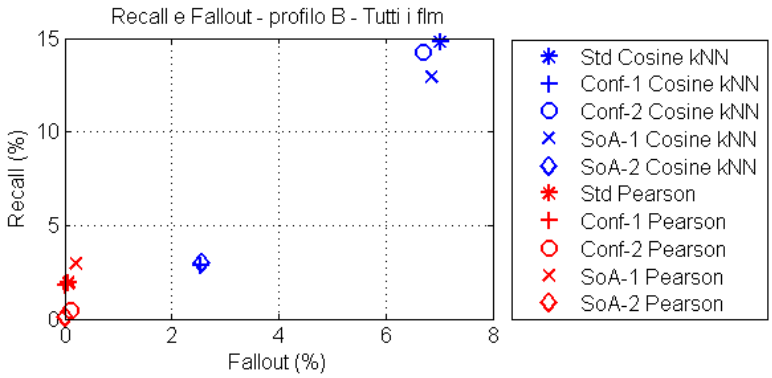


Figura 7.4: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix (ridotto), tutti i film

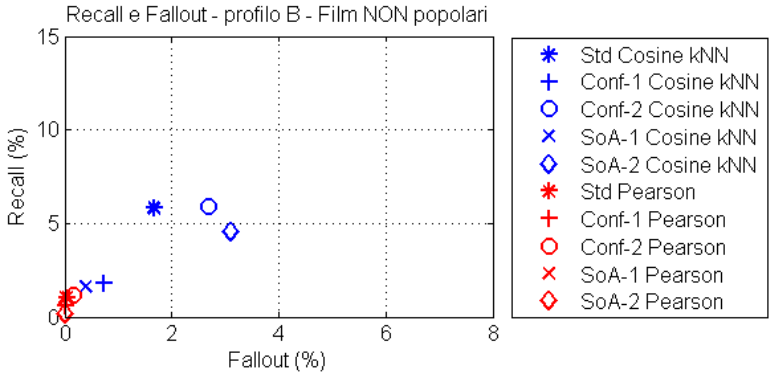


Figura 7.5: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix (ridotto), film non popolari

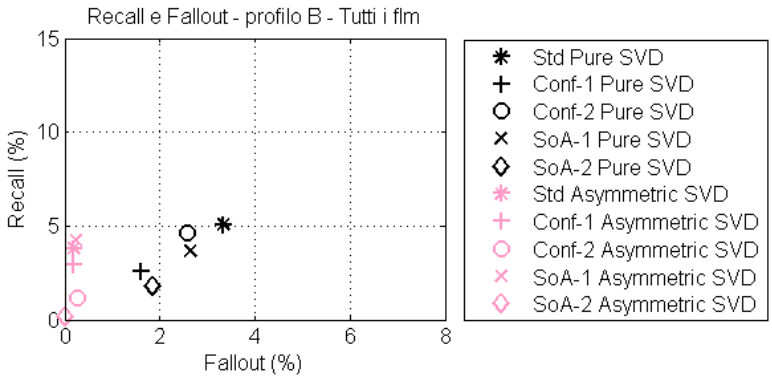


Figura 7.6: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix (ridotto), tutti i film

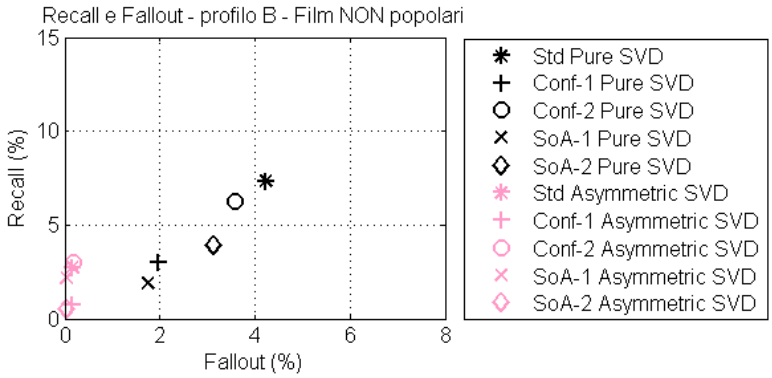


Figura 7.7: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix (ridotto), film non popolari

tato un altro metodo, che restituisse risultati in un tempo accettabile: per ogni profilo utente è stato scelto di testare una certa percentuale di voti appartenenti ad una matrice di test (urmProbeSet), che contiene circa 1,5 milioni di *rating* equamente distribuiti tra gli utenti e i film, compresi in un range tra 1 e 5 e ottenuta dalla matrice URM originale di *Netflix*, che è stata comunque usata per creare il modello e per ricavare i profili utente. Per ottenere dei risultati il più completi possibile, è stato scelto di aggiungere 2 nuovi profili utenti ai 3 già adottati per il resto dei test, in quanto ci è sembrato troppo limitativo fermarci ad analizzare utenti che avessero visto al massimo venti film; i profili aggiunti sono:

- Profilo D: utenti che hanno visto e votato da 21 a 30 film.
- Profilo E: utenti che hanno visto e votato da 31 a 40 film.

Il numero di *rating* di questa matrice è mostrato in figura 7.8: sulle ascisse sono presenti i 5 profili utente, mentre sulle ordinate il numero di film che rientrano in una delle 4 categorie: in nero sono presenti i voti della matrice urmProbeSet con *rating* pari a 5 e in rosso sempre i voti pari a 5 ma soltanto appartenenti ai film non popolari; in blu i voti con *rating* pari a 1 appartenenti a tutti i film e in rosa solo quelli appartenenti ai film non popolari, sempre con *rating* uguale a 1. Data la diversità presente tra i numeri di *rating*, per avere risultati confrontabili è stato scelto di testare circa 500 voti per ogni profilo, per ognuna delle 4 categorie: i voti pari a 5 sono stati usati per testare la *Recall*, mentre quelli pari a 1 per calcolare la

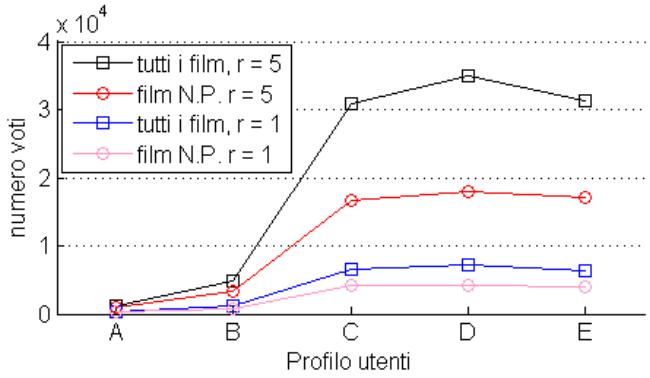


Figura 7.8: Distribuzione dei rating della matrice urmProbeSet tra i diversi profili utente.

Fall-out. I cinque modi in cui *Recall* e *Fall-out* sono state calcolate sono gli stessi usati nei test precedenti, ossia standard, con i 2 metodi da noi ideati e con i due trovati in letteratura per il calcolo della confidenza.

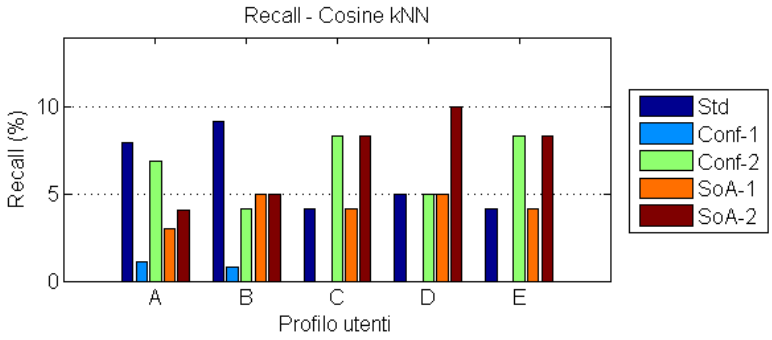


Figura 7.9: Recall dell'algoritmo Item-Item Cosine kNN, Netflix.

Nelle immagini 7.9 e 7.10, sono mostrati rispettivamente i valori di *Recall* per tutti i film del dataset e per i soli film non popolari, relative all'algoritmo Item-Item Cosine kNN. Le figure mostrano i valori di recall in punti percentuali (sulle ordinate) suddivisi per profilo utente (sulle ascisse); per ogni profilo è stata calcolata la *Recall* con i 5 metodi descritti in precedenza: in blu la *Recall* standard, in azzurro quella calcolata tramite il metodo della Confidenza-1, in verde tramite Confidenza-2 e in arancione e marrone le *Recall* calcolate con i metodi trovati in letteratura, che sono rispettivamente relativi al numero di *rating* ottenuti da un film e alla varianza dei suoi voti. Analizzando la figura 7.9 si può notare come la *Recall* SoA-2 aumenti al-

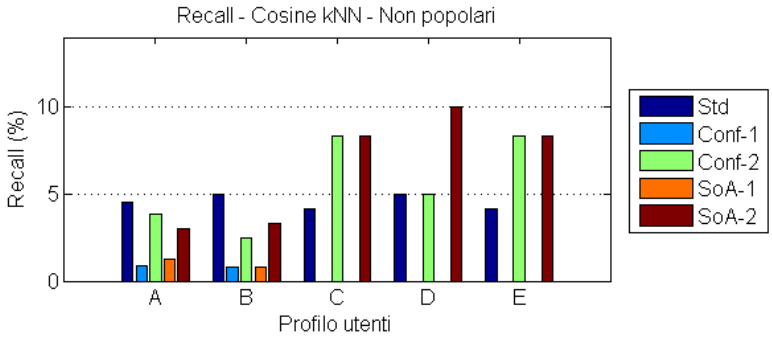


Figura 7.10: Recall dell'algoritmo Item-Item Cosine kNN, film non popolari, Netflix.

l'aumentare del numero di film visti dagli utenti, ossia passando dal profilo A al profilo D, con un sola eccezione per il profilo E in cui questo andamento crescente si interrompe. Per quanto riguarda la *Recall* calcolata con il metodo della Confidenza-1, si può notare che i risultati sono pessimi per ogni profilo utente, soprattutto per i profili degli utenti che hanno visto più di 10 film, per i quali questo valore è nullo, mentre con il secondo metodo per calcolare la confidenza da noi creato, Confidenza-2, si ottengono risultati migliori rispetto a quelli ottenuti con il metodo standard per i profili C, D ed E, ossia questo metodo (adottato con questo algoritmo) dà risultati migliori se calcolato su profili di utenti che hanno visto molti film. La confidenza intesa come numero di *rating* ottenuti da un film, ossia la *Recall* SoA-1, è nulla o quasi per tutti i profili della figura 7.10, relativa appunto ai soli film non popolari.

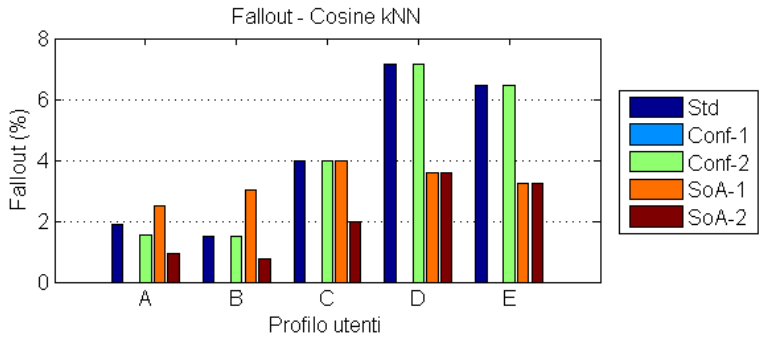


Figura 7.11: Fall-out dell'algoritmo Item-Item Cosine kNN, Netflix.

In figura 7.11 sono mostrati invece i risultati della *Fall-out*, sempre rispettivi all'algoritmo Item-Item Cosine kNN: questi risultati sono mostrati solo per tutti i film del dataset e non più anche per i soli non popolari, in quanto questi valori sono quasi tutti nulli e il mostrarli in figura non avrebbe avuto molto senso: in questo caso i risultati migliori si ottengono con il metodo Confidenza-1, in quanto ottiene valori di *Fall-out* nulli, seguito dal metodo SoA-1 che ottiene valori molto bassi.

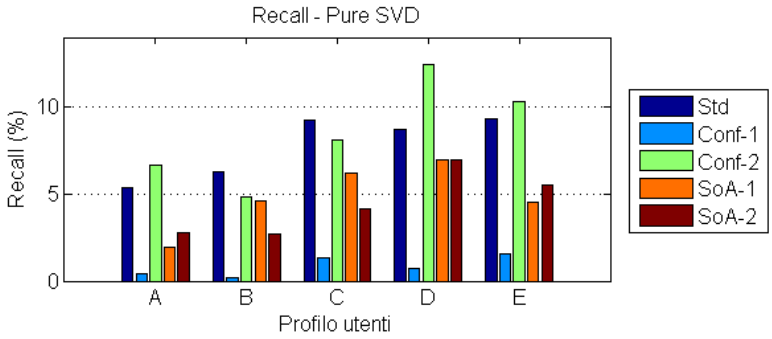


Figura 7.12: Recall dell'algoritmo Pure SVD, Netflix.

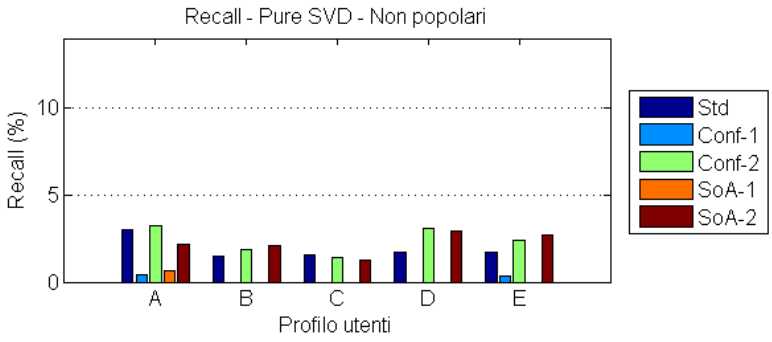


Figura 7.13: Recall dell'algoritmo Pure SVD, film non popolari, Netflix.

La figura 7.12 è riferita invece all'algoritmo Pure SVD: in questo caso il metodo migliore risulta essere quello legato al calcolo della *Recall* tramite Confidenza-2, che in 3 casi su 5 (profili A, D ed E) ottiene un valore maggiore rispetto al metodo standard. I metodi basati sul calcolo della confidenza SoA-1 e SoA-2 ottengono percentuali di *Recall* sempre inferiori a quelle ottenute tramite metodo standard o Confidenza-2, mentre ancora una volta il peggior metodo si dimostra essere quello legato alla Confidenza-1. Per

quanto riguarda i soli film non popolari (figura 7.13) si può notare come la *Recall* calcolata tramite SoA-1 sia quasi sempre nulla (tranne per il profilo A, dove ottiene un valore comunque molto basso, inferiore all'1%), così come la *Recall* calcolata in base alla Confidenza-1; i metodi migliori si dimostrano essere ancora una volta il metodo standard e Confidenza-2, anche se i loro valori percentuali si riducono di molto (per alcuni profili di un fattore 4) passando dall'analizzare tutti i film, ad analizzare solo i non popolari. Per quanto riguarda la *Fall-out* calcolata su tutti i film (figura 7.14), si può notare come, per i profili di utenti che hanno visto un buon numero di film (profili C, D ed E), i metodi Confidenza-2, SoA-1 e SoA-2 peggiorino sempre i risultati ottenuti tramite il metodo standard: in questo caso il metodo migliore è sicuramente Confidenza-1, dando valori molto bassi di *Fall-out*, a volte addirittura nulli (profili A e D). La situazione cambia analizzando solo i film non popolari (figura 7.15): per i profili di utenti che hanno visto molti film, i metodi tendono ad eguagliarsi dando tutti ottimi valori di *Fall-out* (prossimi a 0%), mentre per i profili A e B il metodo migliore è, come succedeva analizzando tutti i film e non solo i non popolari, quello basato su Confidenza-1, seguito da SoA-1.

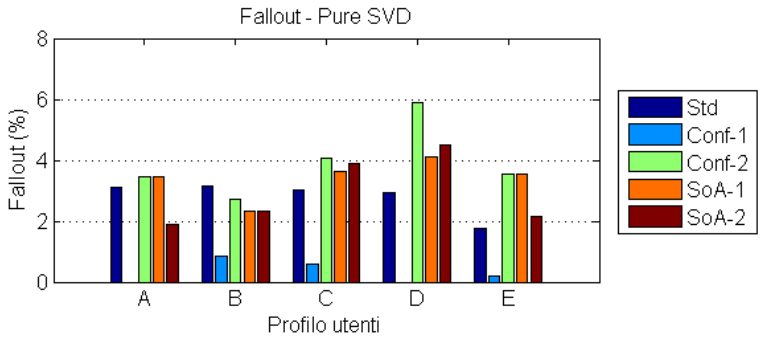


Figura 7.14: Fall-out dell'algoritmo Pure SVD, Netflix.

Una situazione completamente diversa si può osservare nella figura 7.16, dove l'algoritmo usato per il calcolo della *Recall* è Asymmetric SVD: in questo caso i risultati migliori sono dati dal metodo SoA-1, basato sul numero di *rating* ricevuti da ciascun film, che supera il valore ottenuto dal metodo standard in 3 casi su 5 (profilo A, D ed E); con questo algoritmo i nostri metodi si sono rivelati essere molto poco efficaci, i cui valori ottenuti sono maggiori soltanto a quelli ottenuti con il metodo SoA-2, che risulta quindi

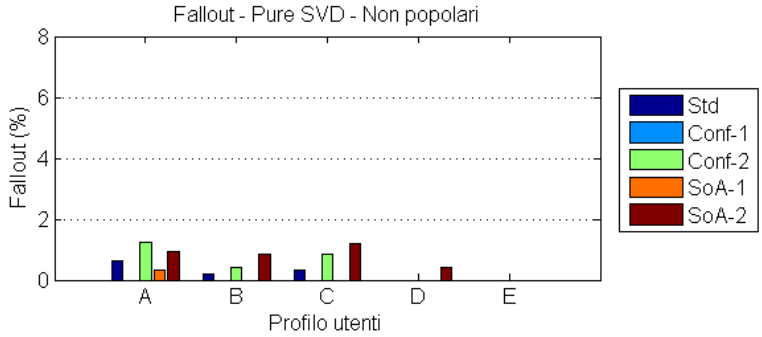


Figura 7.15: Fall-out dell'algoritmo Pure SVD, film non popolari, Netflix.

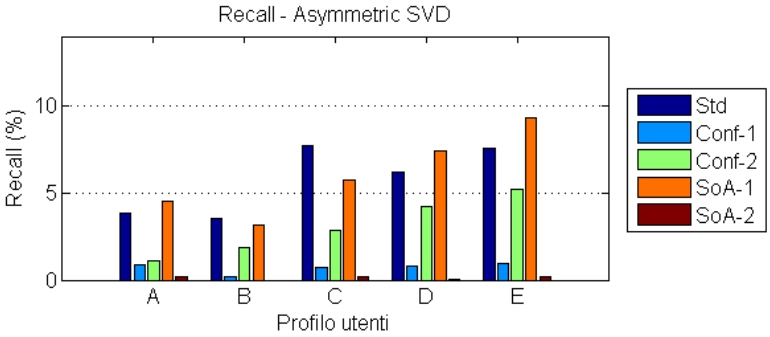


Figura 7.16: Recall dell'algoritmo Asymmetric SVD, Netflix.

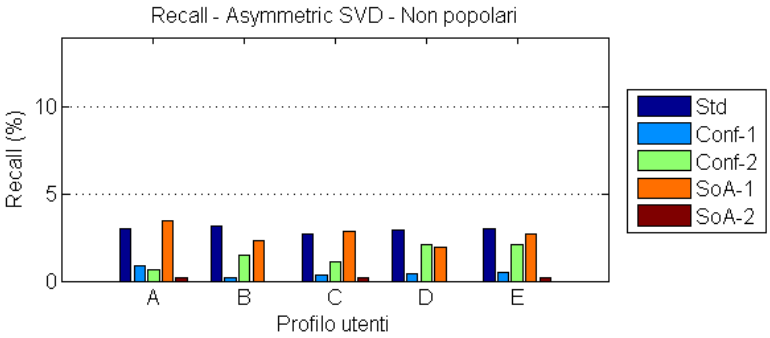


Figura 7.17: Recall dell'algoritmo Asymmetric SVD, film non popolari, Netflix.

essere il peggiore con tutti i profili. Analizzando solo i film non popolari (figura 7.17), il metodo basato sul numero di *rating*, continua inaspettatamente a dare buoni risultati, in alcuni casi migliori di quelli ottenuti con il metodo standard: con questo algoritmo quindi la situazione che emerge dalla comparazione degli algoritmi rimane pressoché invariata. Questo algoritmo ha anche i migliori risultati dal punto di vista della *Fall-out*, calcolata sia analizzando tutti i film, sia analizzando i soli film non popolari, con valori che rimangono sempre inferiori allo 0,4%: confrontare i diversi metodi non ha molto senso in quanto sono tutti ottimi risultati e la leggerissima differenza tra loro può essere trascurata.

Per quanto riguarda l'algoritmo Item-Item Pearson, abbiamo ottenuto dei dati un po' particolari: per i profili C, D ed E, la *Recall* calcolata con il metodo standard, con il metodo legato alla Confidenza-1 e con quello legato al valore della varianza dei *rating* (SoA-2) è risultata nulla: questo fatto ci ha incuriosito e portato a fare dei test su questi profili, per verificare l'attendibilità dei risultati ottenuti. Abbiamo inizialmente selezionato gli utenti della matrice URM appartenenti ai profilo C, D ed E, ossia quegli utenti che hanno visto e dato un voto ad un numero di film compresi tra 11 e 40. Per questi utenti è stato cercato nella matrice urmProbeSet (che ricordiamo contiene dei voti di prova, da usare per testare la *Recall*) un *rating* pari a 5: l'abbiamo rimosso dal profilo utente, abbiamo calcolato una nuova lista di raccomandazione e siamo andati a cercare il film di test in questa lista: abbiamo notato subito che questi film testati non erano presenti nella Top5 e addirittura non erano quasi mai presenti neppure nella lista Top100, pur avendo un *rating* predetto molto alto, spesso prossimo a 5: questo avviene perché nella Top5 erano presenti film con un *rating* predetto addirittura maggiore di 5, a volte prossimo a 6. Sono riportati in seguito un paio di esempi di risultati ottenuti:

- Utente 203073
  Numero film visti: 25 (profilo D) Film presente nella urmProbeSet: "Pay It Forward" con voto pari a 5
  Posizione nella lista di raccomandazione dopo la rimozione del film dal profilo: 483
  rating predetto per il film: 4,65 *rating* predetto per il primo film nella lista di raccomandazione: 5,64

- Utente 83473

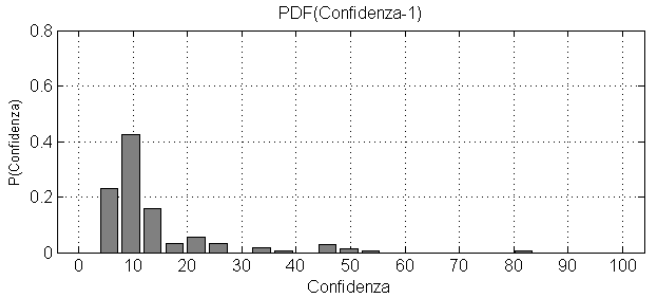
Numero film visti: 16 (profilo C) Film presente nella urmProbeSet: "Law & Order: special victim unit: the first year" con voto pari a 5
Posizione nella lista di raccomandazione dopo la rimozione del film dal profilo: 170
rating predetto per il film: 4,97 *rating* predetto per il primo film nella lista di raccomandazione: 5,73

Ci siamo poi concentrati sugli utenti di profilo C e abbiamo calcolato la *Recall* usando gli altri 2 metodi che hanno dato risultato nullo: Confidenza-1 e SoA-2. In questo caso, dopo aver rimosso dal profilo utente il voto di test, è stata calcolata una lista Top20, è stata ordinata in base ai valori di Confidenza-1 per il primo test e di SoA-2 per il secondo e sono stati tenuti i primi 5 valori ottenuti: anche in questo caso, il film tolto non era mai presente. Sempre manualmente abbiamo calcolato questi valori di Confidenza-1 e SoA-2, per scoprire che erano effettivamente molto bassi. E' inoltre emerso che il film di test, ossia quello che speriamo di ritrovare nella lista di raccomandazione, per oltre il 90% dei casi non risulta essere presente nella Top20. Gli unici valori di *Recall* diversi da 0 per i profili C, D ed E, sono stati ottenuti con i metodi Confidenza-2 e SoA-1 (rispettivamente circa 6% e 3%): abbiamo quindi deciso di calcolare la funzione di densità di probabilità per i 4 metodi usati per calcolare la *Recall* (Confidenza-1, Confidenza-2, SoA-1 e SoA-2) relativi ai film che risultassero presenti nella Top20 degli utenti di questi profili, dopo essere stati rimossi dal profilo utente in quanto presenti nella urmProbeSet. I risultati ottenuti sono mostrati nella figura 7.18, relativa ai test effettuati sul profilo C: nella figura (a) si può notare come i valori di Confidenza-1 dei film analizzati siano bassissimi: questo significa che per quasi tutti gli utenti analizzati, basta togliere dal loro profilo il film visto più simile a quello di cui si vuole calcolare il valore di confidenza affinchè questo sparisca dalla lista di raccomandazione e quindi il valore di *Recall* 0% è giustificato. E' giustificato anche l'altro valore di *Recall* nullo, ossia relativo al metodo SoA-2: dalla figura (d) si nota come la confidenza basata sulla varianza dei *rating* sia nella maggior parte dei casi minore del 50%, e questo fa sì che il film non venga selezionato tra quelli con la varianza minore. Dalle figure (b) e (c) si può notare che anche i valori non nulli di *Recall* sono giustificati: per quanto riguarda la Confidenza-2 (figura (b)) si può notare che nonostante ci sia un'alta percentuale di film con un valore molto basso, ci sono anche circa il 25% di film che hanno ottenuto un valore

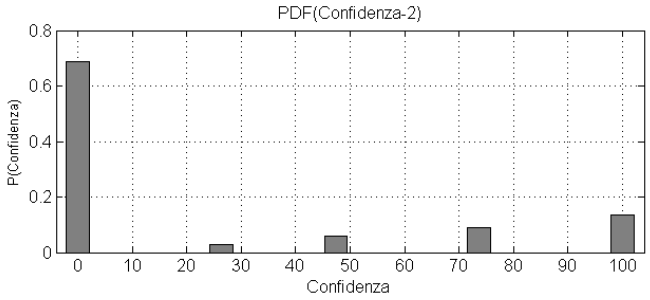
di confidenza maggiore o uguale a 75%, e questo spiega il valore di *Recall* ottenuto con questo metodo pari al 6% circa. Per quanto riguarda la figura (c), relativa al metodo SoA-1, si può notare che anche in questo caso il valore di *Recall* maggiore di zero è giustificato, in quanto sono presenti molti film con un alto valore di confidenza (dato appunto dal numero di *rating* ottenuti da questi film. Nella figura 7.19 sono invece mostrate le funzioni di ripartizione della confidenza calcolata con i 4 metodi. Per quanto riguarda la *Fall-out* calcolata con l'algoritmo di Pearson invece non abbiamo notato anormalità, al contrario, abbiamo ottenuto risultati molto bassi e tra di loro molto simili, con tutti i metodi adottati.

Nelle figure 7.20 e 7.21 è mostrato un confronto tra *Recall* e *Fall-out* calcolate rispettivamente su tutti i film e soltanto su quelli non popolari: gli algoritmi usati sono Pure SVD (in nero) e Asymmetric SVD (in rosa), mentre i valori sono relativi agli utenti di profilo C. Ognuno dei simboli presenti nelle figure rappresenta un diverso metodo per il calcolo delle due metriche di qualità: ad esempio in figura 7.21 si può osservare che per quanto riguarda i soli film non popolari, Asymmetric SVD ha valori di *fall-out* prossimi a zero indipendentemente dal metodo utilizzato, quindi per la valutazione di questi ultimi ci si può basare sul solo valore di *Recall*: si nota che il valore più alto si è ottenuto con il metodo SoA-1, che risulta maggiore anche del metodo standard, mentre il risultato più scadente è dato dall'altro metodo trovato in letteratura per il calcolo della confidenza, SoA-2, relativo alla varianza dei *rating* ottenuti dai film.

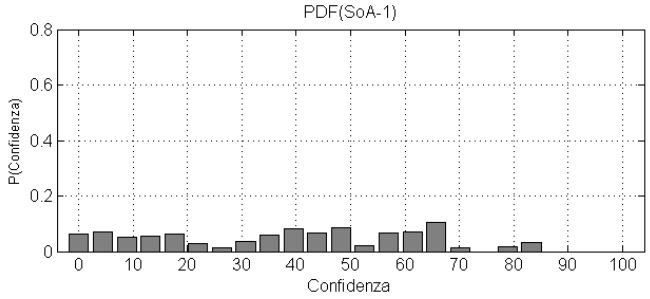
Nelle tabelle 7.1 e 7.2 sono mostrati i valori di *Recall* e *Fall-out* relativi ai 4 algoritmi utilizzati testando solo gli *item* non popolari, rispettivamente per gli utenti di profilo B e di profilo D. E' stato scelto di considerare solo i film non popolari in quanto, come spiegato in [5], raccomandare *item* popolari non porta benefici né agli utenti, né ai fornitori del servizio di raccomandazione, mentre raccomandare film non popolari porta a stupire l'utente e ad aumentare la *novelty* (capitolo 4.1). Sulle righe delle tabelle sono presenti i 4 algoritmi utilizzati, mentre sulle colonne i valori di *Recall* e *Fall-out* divisi nei 5 metodi da noi adottati; ad esempio in figura 7.1 è mostrato che per gli utenti di profilo B, con l'algoritmo Asymmetric SVD il valore di *Recall* ottenuto più alto è relativo al metodo standard, con un valore pari a 3,1447%.



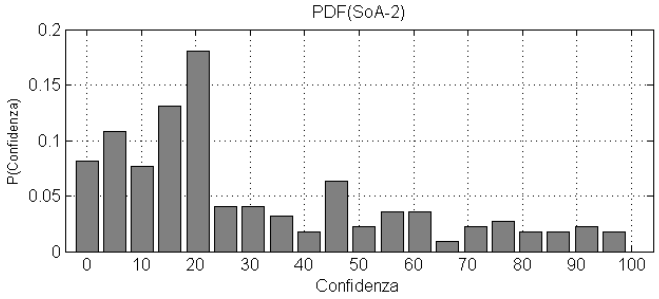
(a) PDF Confidenza-1



(b) PDF Confidenza-2

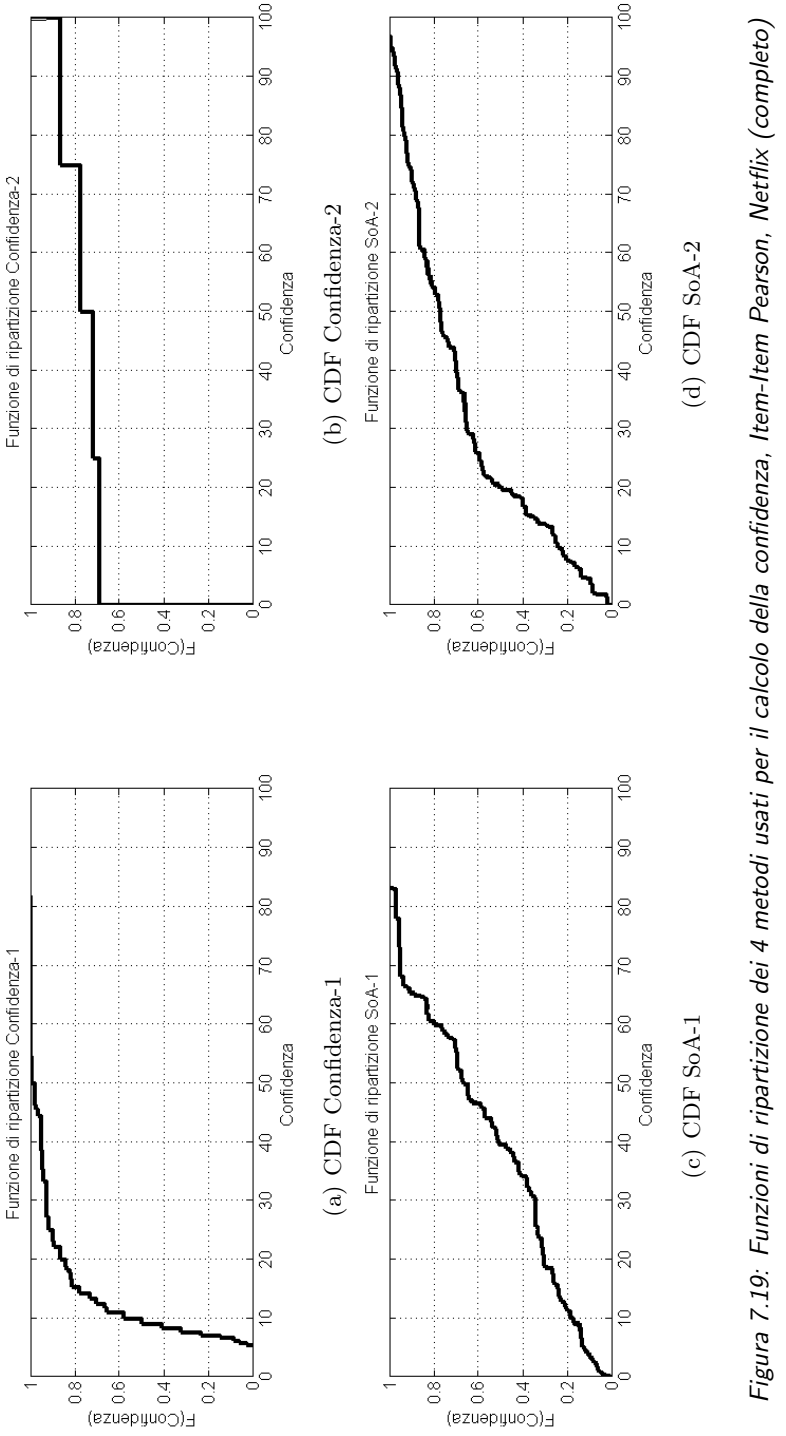


(c) PDF SoA-1



(d) PDF SoA-2

Figura 7.18: Funzioni di densità di probabilità dei 4 metodi usati per il calcolo della confidenza, Item-Item Pearson, Netflix (completo)



(a) CDF Confidenza-1 (b) CDF Confidenza-2

(c) CDF SoA-1 (d) CDF SoA-2

Figura 7.19: Funzioni di ripartizione dei 4 metodi usati per il calcolo della confidenza, Item-Item Pearson, Netflix (completo)

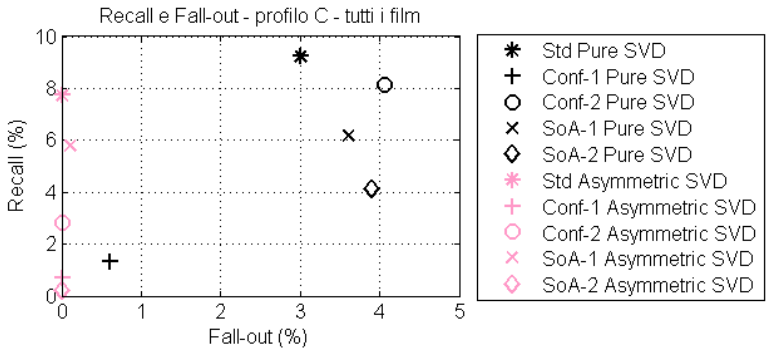


Figura 7.20: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix, tutti i film

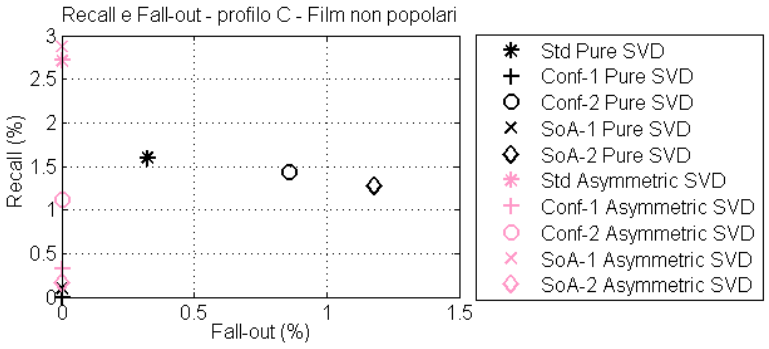


Figura 7.21: Recall e fall-out calcolate con i diversi metodi di confidenza, dataset di Netflix, film non popolari

	Recall					Fall-out				
Metodo:	**Base**	**Conf-1**	**Conf-2**	**SoA-1**	**SoA-2**	**Base**	**Conf-1**	**Conf-2**	**SoA-1**	**SoA-2**
Cosine kNN	5,2566	0,8333	2,5112	0,8333	3,3333	0	0	0	1,5152	0
Pure SVD	1,4675	0	1,8868	0	2,0964	0,2096	0	0,4193	0	0,8386
Asymmetric SVD	3,1447	0,2096	1,4675	2,3061	0	0,2096	0	0	0	0
Pearson	1,6667	0,8333	3,3333	2,5	0,8333	0,7576	0,7576	0,7576	0	0,7576

Tabella 7.1: Recall e Fall-out calcolate con i 5 metodi proposti, profilo B, Netflix (completo), film non popolari

	Recall					Fall-out				
Metodo:	**Base**	**Conf-1**	**Conf-2**	**SoA-1**	**SoA-2**	**Base**	**Conf-1**	**Conf-2**	**SoA-1**	**SoA-2**
Cosine kNN	5,1565	0	5,0522	0	10,1005	0	0	0	0	0
Pure SVD	1,676	0	3,0726	0	2,933	0	0	0	0	0,3922
Asymmetric SVD	2,933	0,419	2,095	1,9553	0	0,1961	0,1784	0,1961	0,3922	0
Pearson	0	0	6,0606	3,0303	0	3,5714	3,148	3,5714	0	3,5714

Tabella 7.2: Recall e Fall-out calcolate con i 5 metodi proposti, profilo D, Netflix (completo), film non popolari

Capitolo 8

Conclusioni e sviluppi futuri

In questo capitolo si andranno ad analizzare i risultati fin qui ottenuti per le spiegazioni e la confidenza nei sistemi di raccomandazione, mostrando quale metodo conviene adottare a seconda dell'algoritmo usato, infine verranno proposti degli spunti per possibili lavori futuri.

8.1 Spiegazioni

Per fornire spiegazioni sulle raccomandazioni proposte agli utenti sono stati adottati due metodi distinti: il primo metodo (metodo normale) consente di dare spiegazioni più complete del secondo (metodo ottimizzato), che d'altra parte impiega molto meno tempo per essere calcolato. Dai valori riassuntivi delle tabelle 5.5 e 5.6 si può però notare come il metodo ottimizzato sia in grado di spiegare una percentuale di film molto alta, simile a quella che è possibile spiegare con il primo metodo, a fronte di un tempo d'esecuzione molto ridotto (tabelle 5.9 e 5.10), a volte anche di 20 volte (per gli utenti di profilo C, usando l'algoritmo Item-Item Cosine kNN, dataset di *Netflix*), dato da un numero di test (tabelle 5.7 e 5.8) anch'essi molto inferiori (con il metodo ottimizzato sono richiesti fino ad un cinquantesimo del numero di test effettuati con il metodo normale). Da questa analisi risulta quindi conveniente spiegare un film consigliato basandosi sui film più simili ad esso presenti nel profilo dell'utente analizzato, e magari (per gli utenti che hanno visto molti film) conviene non fermarsi a rimuoverne soltanto tre per cercare di spiegarlo, ma proseguire fino a cinque, in modo da abbassare ulteriormente la percentuale di film che non è possibile spiegare, che con il metodo ottimizzato dava valori maggiori per gli utenti di profilo C.

8.2 Confidenza

Per il calcolo della confidenza sono stati adottati 4 metodi diversi: due proposti in questo lavoro e basati sulle perturbazioni forzate del profilo utente e due noti in letteratura, che considerano la confidenza come numero di *rating* ottenuti da un film (SoA-1) e come varianza di questi *rating* (SoA-2). Con questi diversi metodi sono poi state calcolate due metriche di qualità delle raccomandazioni, *Recall* e *Fall-out*, confrontando i valori ottenuti con il calcolo standard di queste metriche. Dall'analisi effettuata risulta che, in generale, non c'è un metodo migliore di un altro per dare un valore di confidenza alle raccomandazioni effettuate, ma questi valori cambiano a seconda dell'algoritmo e dal dataset utilizzati; basandosi sui valori ottenuti analizzando i soli film non popolari, possiamo affermare che per quanto riguarda il dataset implicito di *MovieRec*, con i tre algoritmi usati si hanno avuto risultati di *Recall* migliori con il metodo basato sul calcolo della Confidenza-2 da noi ideato: con ogni profilo utente questo metodo infatti ottiene in media valori maggiori anche rispetto al metodo standard, ed è quindi consigliato il suo uso nel calcolo della confidenza dei film raccomandati. Per quanto riguarda il dataset di Netflix, i risultati cambiano a seconda dell'algoritmo utilizzato: con Item-Item Cosine kNN il metodo basato sulla Confidenza-2 in media dà risultati di *Recall* più alti rispetto agli altri metodi, ma il metodo standard rimane il migliore per quanto riguarda il valore di *Fall-out*. Con l'algoritmo Pure SVD la situazione è molto simile, in quanto ancora una volta i due metodi risultano essere i migliori: il metodo standard ha valori di *Recall* più alti, ma il metodo Confidenza-2 ha valori di *Fall-out* più bassi; anche in questo caso quindi i due metodi si equivalgono, e anche con l'algoritmo Asymmetric SVD la situazione non cambia. Dei risultati differenti si hanno con l'algoritmo di Pearson, che per questo motivo è stato analizzato a parte: in questo caso i valori ottenuti sono tutti bassissimi, e quindi non è possibile dire quale metodo sia migliore per il calcolo della confidenza. Riassumendo risulta quindi che il nostro metodo, Confidenza-2, tra i metodi alternativi proposti è decisamente il migliore, in quanto spesso ottiene valori di *Recall* simili (se non maggiori) a quelli ottenuti con il metodo standard, a fronte di una *Fall-out* comunque accettabile.

8.3 Sviluppi futuri

Una limitazione di questo lavoro è il fatto di essere concentrato quasi esclusivamente su algoritmi di tipo collaborativo, fatta eccezione per l'uso dell'algoritmo LSA Cosine (di tipo *Content-Based*) con il dataset di *MovieRec*: questo perché per usare un algoritmo *Content-Based* è necessario avere a disposizione la matrice ICM, che è solitamente calcolata del gestore di servizi di IPTV, e noi non siamo in possesso di questa matrice per il dataset di *Netflix*: un possibile lavoro futuro potrebbe quindi essere il calcolo della confidenza, con i metodi ideati, applicato a questa tipologia di algoritmi, oltre al calcolo delle spiegazioni. Un altro possibile lavoro è l'applicazione dei nostri studi in un caso reale, mostrare cioè agli utenti i risultati da noi ottenuti ed analizzare il loro comportamento a fronte delle raccomandazioni date con le relative spiegazioni e il valore di confidenza, per vedere se effettivamente questi valori invogliano l'utente a fare maggior uso del sistema, aumentando anche il grado di fiducia che hanno in esso.

Bibliografia

[1] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. Towards More Confident Recommendations: Improving Recommender Systems Using Filtering Approach Based on Rating Variance. *IEEE Trans. on Knowl. and Data Eng.*, pages 734–749, 2005.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.*, pages 734–749, June 2005.

[3] Mustafa Bilgic. Explaining Recommendations: Satisfaction vs. Promotion. In *In Proceedings of Beyond Personalization 2005, the Workshop on the Next Stage of Recommender Systems Research*, pages 13–18, 2005.

[4] Elisa Campochiaro, Paolo Cremonesi, and Roberto Turrin. Analysis of Recommender Systems based on implicit datasets. *Technical report*, 2008.

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[6] Paolo Cremonesi and Roberto Turrin. Analysis of cold-start recommendations in IPTV systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 233–236, New York, NY, USA, 2009. ACM.

[7] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Seman-

tic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, pages 61–70, 1992.

[9] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, New York, NY, USA, 2000. ACM.

[10] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, pages 5–53, 2004.

[11] Parry Husbands, Horst Simon, and Chris H. Q. Ding. *On the use of the singular value decomposition for text retrieval*, pages 145–156. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[12] Jens F. Jensen. Interactive television - a brief media history. In *Proceedings of the 6th European conference on Changing Television Environments*, EUROITV '08, pages 1–10, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *IJCAI*, pages 1137–1145, 1995.

[14] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[15] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4:1:1–1:24, January 2010.

[16] David McSherry. Explanation in Recommender Systems. *Artif. Intell. Rev.*, 24:179–197, October 2005.

[17] Michael Pazzani and Daniel Billsus. Content-Based Recommendation Systems. pages 325–341. 2007.

[18] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[19] Upendra Shardanand and Patti Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[20] Sean Mcnee Shyong, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. Confidence Displays and Training in Recommender Systems. In *Proceedings of the 9th IFIP TC13 International Conference on HumanComputer Interaction (INTERACT*, pages 176–183. IOS Press, 2003.

[21] Nava Tintarev. Explanations of recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 203–206, New York, NY, USA, 2007. ACM.

[22] Nava Tintarev and Judith Masthoff. A Survey of Explanations in Recommender Systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 801–810, Washington, DC, USA, 2007. IEEE Computer Society.

[23] Nava Tintarev and Judith Masthoff. Effective explanations of recommendations: user-centered design. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 153–156, New York, NY, USA, 2007. ACM.

[24] Nava Tintarev and Judith Masthoff. Designing and Evaluating Explanations for Recommender Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer US, 2011.

[25] Roberto Turrin and Paolo Cremonesi. Controlling Consistency in Top-N Recommender Systems. In *ICDM2010*, 2010.

[26] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 47–56, New York, NY, USA, 2009. ACM.

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