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

RACCOMANDAZIONI SU CANALI DI DISTRIBUZIONE UNIDIREZIONALI

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

Tesi di Laurea di:
Ettore Cremonesi
Matricola n. 719831

Anno Accademico 2010-2011

Sommario

In un contesto televisivo dove la quantità di contenuti proposta aumenta quotidianamente in modo consistente, si è fatta largo la necessità tra i fornitori di servizi TV di aiutare gli utenti nella ricerca di trasmissioni rilevanti adottando sistemi di raccomandazione capaci di suggerire un insieme personalizzato di programmi potenzialmente di loro interesse.

Nella letteratura scientifica sono molteplici gli approcci proposti per generare raccomandazioni agli utenti. Gli algoritmi più utilizzati, noti come filtri collaborativi, prevedono l'acquisizione e l'elaborazione di informazioni relative alle preferenze della comunità di utenti che utilizzano il sistema.

Non tutti i canali di distribuzione dei contenuti televisivi consentono però, al fornitore del servizio, di raccogliere le preferenze degli utenti. Ad esempio le piattaforme digitale terrestre e satellitare non dispongono del canale di ritorno dall'utente al sistema, negando la possibilità di collezionare tali informazioni.

Questo lavoro di tesi si propone di studiare i sistemi di raccomandazione in uno scenario senza canale di ritorno, dominio finora inesplorato della letteratura.

Verrà introdotta e valutata l'idea innovativa di utilizzare dati esterni raccolti da un fornitore di contenuti operante su un canale bidirezionale distinto, come ad esempio un portale Web, per compiere raccomandazioni sul canale unidirezionale in cui tali informazioni non sono invece presenti.

Indice

Capitolo 1

Introduzione

Con la recente diffusione di tecnologie quali il digitale terrestre e l'IPTV, la proposta televisiva è cresciuta a tal punto da rendere difficoltosa la ricerca dei contenuti interessanti per l'utente. E' perciò fondamentale che tali tecnologie supportino l'utente aiutandolo a trovare film, trasmissioni o canali televisivi rilevanti, ad esempio selezionando e consigliando alcuni contenuti.

Inizialmente i contenuti più significativi erano selezionati e proposti agli utenti senza considerarne i gusti specifici, anche se era chiaro come un suggerimento mirato per ciascun utente avrebbe fatto maggior breccia nei suoi desideri [8]. La nascita dei sistemi di raccomandazione nei primi anni anni '90 ha segnato una svolta nel campo della personalizzazione dei contenuti [26]. Tali sistemi raccomandano contenuti personalizzati in base agli interessi di ogni singolo utente.

Per collezionare le informazioni relative alle preferenze degli utenti è necessaria la presenza di un canale di comunicazione bidirezionale tra il fornitore del servizio e l'utente, così che il primo possa erogare i contenuti e le raccomandazioni mentre il secondo possa comunicare al provider, implicitamente o esplicitamente, il gradimento di quanto utilizzato tramite un canale di ritorno (feedback).

I sistemi di raccomandazione sono stati impiegati da sempre come strumento per massimizzare i profitti di piattaforme di e-commerce [38] e di online advertising [27], grazie alla facilità di collezionare le informazioni relative agli utenti. Questa semplicità è dovuta alle caratteristiche intrinseche di Internet che permette uno scambio vicendevole di informazioni tra l'utente e la piattaforma online.

Tutti i sistemi proposti o sviluppati fino ad oggi poggiano tuttavia sull'assunzione implicita che sia presente un canale di feedback. Questa ipotesi non è però valida per un insieme di canali di distribuzione unidirezionali ampiamente diffusi, quali la piattaforma televisiva digitale terrestre [24], le trasmissioni satellitari e quelle analogiche via cavo.

La tesi si pone l'obiettivo di valutare la realizzazione di sistemi di raccomandazione per contesti televisivi basati su canali di distribuzione uniderezionali, ovvero dove non è presente alcun flusso di informazioni dall'utente verso il fornitore del servizio.

1.1 Contributo della tesi

In questo lavoro di tesi è stato considerato un dominio televisivo in cui la mancanza di un canale di feedback impedisce al fornitore del servizio di collezionare le preferenze degli utenti. Sono stati quindi proposti e valutati diversi approcci ispirati ai metodi classici trattati in letteratura, allo scopo di valutare se fossero implementabili nel nuovo contesto con canale unidirezionale e con quali limitazioni.

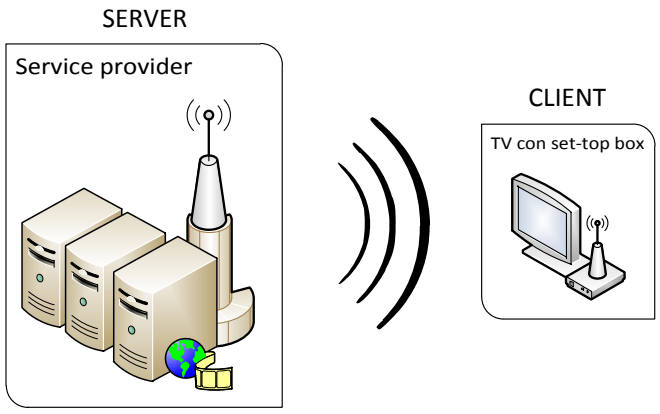


Figura 1.1: Scenario di trasmissione su canale broadcast unidirezionale

Ipotizziamo quindi, come esemplificato in Figura 1.1, che ci sia un server in cui siano memorizzati i contenuti (es. film) con le relative informazioni (es. titolo) e un client (ad esempio un set-top box o decoder) che permetta all'utente di visualizzare il materiale trasmesso dal server. Le preferenze dell'utente possono essere memorizzate nel client, che non può in alcun modo comunicare attivamente con il server. In un classico contesto dove non ci siano vincoli legati al canale il server colleziona le preferenze degli utenti e, quando richiesto, determina la lista di contenuti da raccomandare e la invia al client, il quale la mostra all'utente. Nel dominio in esame questo approccio non è possibile, sarà pertanto responsabilità del client determinare la lista di raccomandazione da presentare all'utente essendo l'unico componente a conoscenza delle sue preferenze.

Il più semplice approccio alle raccomandazioni in questo ambiente è quello basato su metodi *non personalizzati*. Ad esempio, si può proporre agli utenti una lista statica di oggetti selezionati da una redazione. Una soluzione di questo tipo è realizzabile nel dominio unidirezionale perché non necessita che il server abbia alcuna informazione relativa agli utenti: il server genera la lista e la invia al client. Si noti che la pubblicità attualmente disponibile sulle piattaforme televisive è di questo tipo.

Una semplice evoluzione dei metodi non personalizzati consiste nell'uso di informazioni *demografiche* relative all'utente, come ad esempio il genere e l'età, così da rendere più mirati i suggerimenti personalizzandoli per categorie di utenti. Ad ogni singola categoria viene assegnata una lista statica così che ogni utente abbia delle raccomandazioni correlate alla propria classe di appartenenza. Il problema che si riscontra utilizzando questa tecnica è che non è possibile calcolare le liste di raccomandazioni lato server, ma è necessario dividere la logica di business spostandone una parte lato client. Il server si occuperà così di inviare un insieme di liste ai client e questi dovranno selezionare quale lista è più adeguata in base alle caratteristiche dell'utente (ipotizzando, realisticamente, che il client abbia tale informazione).

Consideriamo ora gli algoritmi di raccomandazione personalizzati, che possono essere distinti in content-based e collaborativi. I metodi che utilizzano informazioni relative al contenuto (es. titolo, attori) degli oggetti da consigliare sono noti come *content-based*. Sfruttando, come nel caso precedente, l'idea di spostare parte della computazione lato client (es. set-top box) è possibile calcolare le raccomandazioni. Infatti lato server viene calcolata la somiglianza tra i diversi oggetti sfruttando le informazioni sul loro contenuto e tale informazione viene inviata al client. Il client può quindi calcolare i contenuti da consigliare andando a selezionare quelli più simili agli oggetti che l'utente ha trovato di gradimento nel passato (ipotizzando che il client abbia tenuto traccia dei contenuti visti in precedenza).

Se gli approcci fin qui proposti sono in grado di funzionare in presenza di un canale unidirezionale con l'apporto di semplici modifiche, la famiglia più nota e diffusa di sistemi di raccomandazione, quelli *collaborativi* [35], non può essere implementata senza una revisione completa del meccanismo di funzionamento. Questi algoritmi selezionano infatti i contenuti da raccomandare sulla base delle preferenze espresse da tutti gli utenti del sistema. L'impossibilità di trasferire le informazioni dall'utente al fornitore del servizio rende perciò impossibile il calcolo di tali informazioni rendendo la soluzione inapplicabile.

Questa limitazione ha portato alla nascita dell'idea innovativa alla base di questo lavoro di tesi, ossia apprendere un modello collaborativo dai dati provenienti

da un fornitore di contenuti che dispone di un canale di distribuzione bidirezionale per poi utilizzarli in fase di raccomandazione presso un fornitore diverso provvisto di canale unidirezionale. La possibilità di portare il modello da un provider ad uno differente è subordinata alla presenza, almeno parziale, di oggetti comuni ai due fornitori (ad esempio un catalogo comune di film). Questa strategia, definita *porting*, si basa sulla congettura che gli utenti di diversi service provider presentino comportamenti simili. Ad esempio se un provider dovesse rilevare che i suoi utenti che apprezzano il film Matrix hanno espresso giudizi positivi anche per Avatar, è lecito aspettarsi che questo fenomeno possa essere osservato in media anche dagli altri provider.

Considerando infine i metodi *ibridi*, che utilizzano diverse strategie eterogenee, essi presentano tutte le criticità dei metodi che incorporano. Ad esempio un metodo che sfrutterà sia informazioni collaborative che content richiederà il porting del modello.

Gli algoritmi collaborativi proposti sono stati implementati e testati per confermare l'effettiva applicabilità del porting del modello. Nella fase preliminare di testing i dataset utilizzati sono stati costruiti artificialmente per valutare come le dimensioni dei dataset e le loro densità possano avere un impatto sulla qualità delle raccomandazioni. Per disporre di valutazioni significative i dataset artificiali sono stati creati a partire da informazioni contenute nel dataset reale Netflix, contenente circa 10 milioni di rating.

In un secondo tempo, diversi metodi (non personalizzati, content-based, collaborativi e ibridi) sono stati testati su una coppia di dataset reali, MovieLens e Yahoo!, scelti per avere una consistente parte degli oggetti in comune, così da sperimentare il porting dei modelli da uno all'altro e viceversa.

Gli algoritmi proposti sono stati valutati con le metriche più diffuse nell'ambito dei sistemi di raccomandazione: recall, fallout e curva di ROC [55]. I risultati ottenuti utilizzando entrambe le tipologie di dataset, artificiali e reali, sono stati discussi e confrontati tra loro al fine di trarne conclusioni generali.

Aspetti innovativi

L'uso di sistemi di raccomandazione in presenza di un canale unidirezionale, che limita quindi il transito di informazioni verso il fornitore del servizio, è un problema finora non affrontato in letteratura. Il valore dell'analisi compiuta in questo ambiente è evidente se si considera l'importanza e la diffusione delle piattaforme televisive digitale terrestre e satellitare che utilizzano proprio canali di distribuzione unidirezionali.

Inoltre l'idea di effettuare il *porting* di un modello, appreso su un insieme di dati per effettuare raccomandazioni su uno differente, è nuova ed assente in letteratura. Il concetto potrà essere esteso ad altri campi per colmare l'assenza di informazioni con modelli appresi su dati provenienti da ambienti diversi ma che descrivono il medesimo fenomeno.

Le idee e le metodologie proposte in questo lavoro di tesi propongono interessanti punti di partenza per futuri studi e prodotti commerciali legati al mondo televisivo.

Risultati rilevanti

I risultati ottenuti in fase di testing mostrano come il porting del modello da un dominio ausiliario (con canale bidirezionale) sia una valida idea per supplire alla mancanza di informazioni nel dominio target (con canale unidirezionale).

Per alcuni algoritmi si è inoltre verificato un fenomeno ancora più interessante, per cui il porting da un dominio ausiliario denso ad uno target sparso ha portato a dei miglioramenti nella qualità delle raccomandazioni. Ad esempio, in caso di porting dal dataset MovieLens a quello Yahoo!, il metodo AsymmetricSVD guadagna il 7.66% in termini di recall e vede ridurre il fallout del 7.35%.

1.2 Struttura della tesi

Il Capitolo 2 dapprima introduce il lettore ai sistemi di raccomandazione per poi analizzarne più nel dettaglio i singoli componenti e le caratteristiche salienti. Vengono infine presentate le quattro principali famiglie di algoritmi esistenti in letteratura: non personalizzati, content-based, collaborativi e ibridi.

Nel corso del Capitolo 3 viene presentato e formalizzato il problema della raccomandazione su un canale unidirezionale e vengono proposte diverse alternative che fanno riferimento alle diverse famiglie dei sistemi di raccomandazione. Inoltre viene compiuta un'analisi sui requisiti hardware necessari all'implementazione reale delle soluzioni. Infine viene discusso il problema della privacy nel campo delle raccomandazioni.

La divisione dei test compiuti, prima su dataset artificiali e poi reali, è discussa nel Capitolo 4, dove si affrontano anche i problemi legati all'ottimizzazione dei singoli metodi implementati.

Il Capitolo 5 presenta i dataset MovieLens e Yahoo! utilizzati nella fase di test su dataset reali. Anche la descrizione del dataset Netflix, utilizzato per i test su dataset artificiali, trova menzione in questo capitolo.

La metodologia utilizzata per svolgere i test, come le metriche utilizzate per valutare gli algoritmi così come le strategie utilizzate per validare i risultati, trova spazio nel Capitolo 6.

Le descrizioni e le discussioni dei risultati ottenuti sui dataset artificiali e reali sono riportate rispettivamente nei Capitoli 7 e 8.

Infine il Capitolo 9 è dedicato a raccogliere le conclusioni di questo lavoro di tesi lasciando possibili spunti per sviluppi futuri.

Capitolo 2

Stato dell'arte

In questo capitolo vengono presentati i sistemi di raccomandazione e la loro architettura generale. In seguito, a partire dal Paragrafo 2.3, vengono introdotte le famiglie principali dei sistemi di raccomandazione: *non personalizzati*, *content-based*, *collaborativi* e *ibridi*. Per ognuna di esse vengono infine presentati gli algoritmi più rappresentativi.

2.1 I sistemi di raccomandazione

La mole di informazioni presenti su Internet ha raggiunto una dimensione tale [21] che è divenuto impossibile trovare un particolare contenuto senza avvalersi di strumenti in grado di selezionare le alternative più rilevanti. Una semplice ricerca sul Web attraverso un motore di ricerca è in grado di generare milioni di risultati, per cui è importante che il servizio sia in grado di riconoscere i contenuti più importanti e di ordinarli di conseguenza [45].

Un sistema di raccomandazione è essenzialmente uno strumento che permette di estrarre gli elementi più rilevanti per l'utente da un vasto insieme di possibili alternative [1]. La scelta di quali risultati presentare all'utente può essere compiuta con le modalità più diverse, ad esempio sulla base di informazioni che tengano traccia dello storico delle ricerche compiute dall'utente piuttosto che analizzando le informazioni relative agli elementi oggetto della ricerca.

I primi sistemi di raccomandazione fecero la loro comparsa all'inizio degli anni '90 [52] come punto di incontro tra diverse discipline tra cui machine learning, scienze sociali ed economiche. Se all'inizio si trattava di pura ricerca scientifica e si indagavano ancora i possibili impatti sociali e gli eventuali risvolti economici [53], al giorno d'oggi i sistemi di raccomandazione sono parte integrante del core

business delle più grandi società di vendita di beni e servizi, come ad esempio Amazon [38] e Google [27].

Sebbene il concetto alla base dei sistemi di raccomandazione sia semplice, sotto la superficie essi celano una struttura complessa ed articolata che nel corso del capitolo verrà sezionata ed analizzata approfonditamente.

2.2 Architettura dei sistemi di raccomandazione

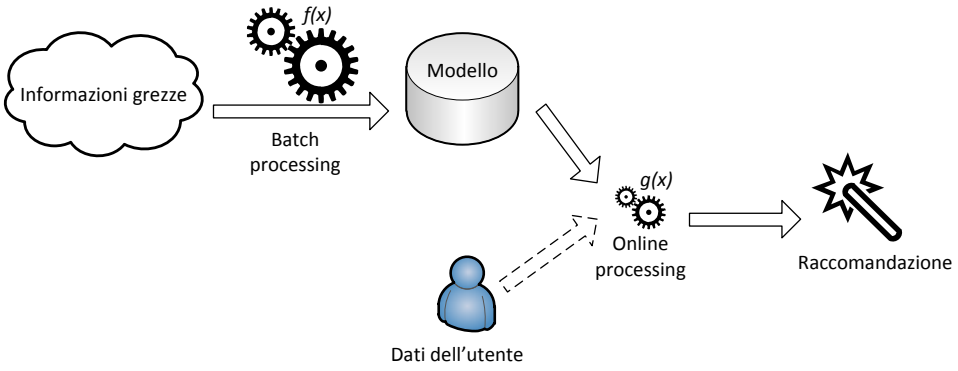


Figura 2.1: Funzionamento di un sistema di raccomandazione

Da un punto di vista generale un sistema di raccomandazione è costituito da un insieme di dati grezzi che descrivono alcuni oggetti[1] e da un algoritmo che è in grado di operare su questi dati stabilendo un ordinamento tra i diversi elementi.

Gli oggetti possono essere ad esempio libri o film mentre i dati grezzi sono informazioni che li riguardano come ad esempio la data di pubblicazione per un libro o gli attori per un film.

L'eventuale aggiunta di informazioni relative all'utente che richiede la raccomandazione consente di personalizzare l'esito in funzione delle caratteristiche disponibili riguardo l'individuo, come ad esempio il genere di film preferito.

Formalmente la raccomandazione personalizzata consiste nella stima della valutazione, o rating, che l'utente assegnerebbe agli item che non ha ancora utilizzato. Siano U l'insieme degli utenti e I l'insieme degli oggetti, il rating espresso per l'item i dall'utente j viene indicato con r_{ij}. L'insieme di tutte le valutazioni del medesimo utente j sono raccolte nel suo profilo utente indicato

[1] Oggetto, elemento ed item sono utilizzati come sinonimi per esprimere l'oggetto della raccomandazione.

come $\boldsymbol{p_j}$. Infine le stime dei rating e del profilo vengono indicate rispettivamente come $\hat{r}_{ij}$ e $\hat{\boldsymbol{p}}_{\boldsymbol{j}}$.

$$\begin{aligned} \forall j \in U \qquad & \hat{\boldsymbol{p}}_{\boldsymbol{j}} = \underset{\hat{r}_{ij}}{\operatorname{argmin}} \sum_{i \in I} |r_{ij} - \hat{r}_{ij}| \\ \forall j \in U \qquad & \boldsymbol{p_j} = \{r_{ij} \quad \forall i \in I\} \end{aligned} \tag{2.1}$$

Nella Formula 2.1 si evidenzia l'obiettivo di stimare il profilo $\hat{\boldsymbol{p}}$ per ogni utente j minimizzando la differenza tra il vero rating r_{ij}, espresso per l'item i dall'utente j, e quello stimato $\hat{r}_{ij}$. Una volta calcolati i rating il sistema ordina gli elementi in funzione della valutazione e li propone così all'utente.

Quando possibile, l'algoritmo di raccomandazione viene suddiviso in due parti distinte dette batch, o offline, e online. La parte batch si occupa di effettuare tutte le computazioni possibili con i dati a disposizione in assenza dell'utente, costruendo di fatto un modello dei dati. Come mostrato in Figura 2.1, una volta giunta la richiesta di raccomandazione da uno specifico utente la parte online dell'algoritmo provvederà a combinare il modello con i dati dell'individuo calcolando la raccomandazione personalizzata.

I dati su cui possono operare i sistemi di raccomandazione sono generalmente organizzati in strutture matriciali, mostrate in Figura 2.2, che si distinguono per la tipologia di dati che possono immagazzinare. Queste matrici sono note come URM (User Rating Matrix), ICM (Item Content Matrix) e UCM (User Content Matrix).

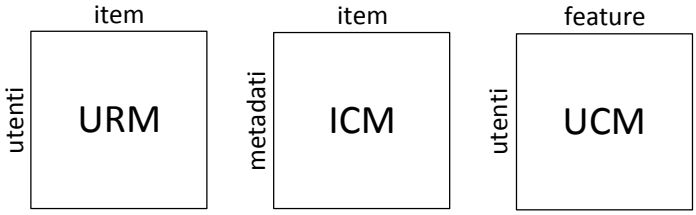


Figura 2.2: URM, ICM e UCM

User Rating Matrix

La URM è una matrice che contiene il gradimento, detto rating, di un particolare elemento da parte di uno specifico utente e questo valutazione può essere implicita oppure esplicita .

É implicita [31] quando l'utente non indica il proprio giudizio sui diversi oggetti e quindi è necessario desumere il gradimento dal solo fatto che ha fruito o meno di un determinato elemento, a tale scopo verrà indicato un valore booleano. Ad esempio un valore pari a 1 nella cella di coordinate (3,7) indicherà che l'utente 3 ha utilizzato l'item 7.

Per contro è esplicita [65] quando un utente esprime il proprio gradimento come scelta tra diverse opzioni possibili, ad esempio valutazioni da 1 a 5 oppure testuali come pessimo, sufficiente ed ottimo. Ad esempio un valore pari 2 su un massimo di 3 nella cella di coordinate (10,6) porterà l'informazione che l'utente 10 ha trovato sufficiente l'item 6.

La scelta del tipo di dataset, cioè dell'insieme dei dati, è vincolata da come vengono collezionate le informazioni. Un dataset implicito è costruito in maniera totalmente trasparente all'utente liberandolo dall'onere di dover esprimere per ogni item il suo personale giudizio e può risultare quindi più user-friendly. Per contro un dataset esplicito contiene molte più informazioni che potrebbero portare a raccomandazione con una qualità superiore. Questo miglioramento teorico della qualità si scontra poi nei casi reali con problematiche dovute al bias introdotto da ciascuno nelle proprie valutazioni per cui un utente potrebbe rivelarsi molto critico oppure troppo magnanimo, alterando di fatto l'affidabilità del modello. A supporto di questa congettura esistono dimostrazioni sperimentali sul fatto che la qualità delle raccomandazioni fornite usando un dataset implicito è assimilabile a quella garantita da un dataset esplicito [14]. Si osservi che le informazioni nella URM potrebbero essere miste, sia esplicite che implicite, per cui è necessario comprendere per ogni caso reale quale sia la miglior metodologia da adottare per compiere le raccomandazioni.

Item Content Matrix

La ICM contiene la rilevanza che ogni metadato ha per ciascun oggetto. Con metadato si intende qualunque termine correlato con l'elemento considerato, ad esempio se gli item fossero film alcuni tra i metadati potrebbero essere l'anno di uscita, il regista e gli attori. Anche parole estratte dalla sinossi del film potrebbero essere usate come metadati poiché alcune caratterizzano il film indicandone ad esempio il contesto storico.

La rilevenza di ciascun metadato può essere espressa, come nel caso della URM, o come semplice valore booleano che ne indica la presenza o meno oppure mediante un coefficiente che ne indica l'importanza per quello specifico oggetto. Questa scelta porta alla creazione di ICM implicite o esplicite.

Si noti che l'eventuale presenza di termini troppo comuni potrebbe rendere tutti gli item simili tra loro. A tal fine è opportuno considerare l'uso di una ICM esplicita per indicare l'importanza dei metadati, così da valorizzare le parole più significative e rendere trascurabile l'influenza dei termini inutili che anzi potrebbero peggiorare la qualità delle raccomandazioni. Una tecnica utilizzata per risolvere questo problema, nota come *tf-idf*, verrà presentata nel Paragrafo 2.5.

User Content Matrix

La UCM è una matrice che contiene le informazioni demografiche relative all'utente. Un esempio di questi dati potrebbe essere l'età o il genere.

Poiché questa matrice non contiene alcun dato riguardante gli item non può essere direttamente impiegata per compiere le raccomandazioni ma piuttosto aggiunge informazioni che possono essere sfruttate in modo diverso. Ad esempio sarebbe possibile valutare la somiglianza di due individui in base all'età e al genere, così da suggerire ad un utente elementi giudicati rilevanti da un altro individuo a lui simile.

Nella realtà queste matrici contengono un numero molto ridotto di informazioni, come è osservabile nel Paragrafo 5.1. Si consideri ad esempio la URM, per ogni utente ci saranno disponibili tante celle quanti sono gli item anche se è impensabile che un individuo possa averli utilizzati tutti. Caso ancora più eclatante è costituito dalla ICM dove per ogni item ci saranno tante celle quanti sono i metadati, alcuni dei quali così specifici che magari apparterranno a solo un elemento.

Inoltre la mancanza di valori all'interno delle matrici potrebbe indicare sia che l'utente non ha ancora sperimentato l'item sia che non intenda farlo. Non potendo sapere il significato di questo vuoto si assume sempre che l'assenza indichi che l'utente non ha usufruito dell'item ma ne è potenzialmente interessato.

In funzione dei dati utilizzati, collaborativi, content-based o demografici, e delle strategie adottate per giungere alla raccomandazione, i sistemi di raccomandazione possono essere catalogati in diverse categorie.

2.3 Classificazione dei sistemi di raccomandazione

Nei sistemi di raccomandazione esistono diverse famiglie di algoritmi raggruppati in base ai dati che utilizzano.

Gli algoritmi più semplici sono quelli non personalizzati [58] che producono raccomandazioni uguali per tutti gli utenti a partire dalla matrice URM. Ad esempio nel caso di compiere raccomandazione nell'ambito cinematografico, un possibile algoritmo non personalizzato potrebbe suggerire la visione dei film più popolari tra gli utenti.

Gli algoritmi personalizzati sono invece tutti quei metodi che utilizzano informazioni relative all'utente e che esprimono il suo interesse per gli item. La struttura che li contiene, detta profilo, è un vettore che dispone di tante celle quanti sono gli oggetti di cui l'utente abbia fruito e contiene per ognuno di essi una valutazione, implicita o esplicita. La famiglia dei metodi personalizzati è a sua volta divisibile in algoritmi content-based, collaborativi e ibridi.

I metodi content-based [48] fanno uso della ICM e propongono item simili a quelli per cui l'utente ha espresso gradimento. Ad esempio se un utente avesse gradito molti film di Steven Spielberg il sistema gli consiglierebbe certamente altre opere del medesimo regista.

Per contro i metodi collaborativi [29] utilizzano la URM e consigliano oggetti ritenuti interessanti per gli utenti con gusti simili ai propri. Se per esempio due utenti apprezzassero gli stessi libri e uno dei due dovesse leggerne uno nuovo, il sistema potrebbe suggerire all'altro il medesimo libro.

Infine gli algoritmi ibridi [5] raggruppano tutti quei metodi che utilizzano contemporaneamente tipi eterogenei di dati fondono insieme algoritmi diversi per tentare di sfruttare i punti di forza di ciascuno.

I vari algoritmi possono essere classificati ortogonalmente anche come model-based [66] e memory-based [52]. I primi sono quelli più diffusi e, come mostrato in Figura 2.1, costruiscono dapprima un modello dei dati per poi operare la raccomandazione in un secondo tempo. I metodi memory-based invece non producono alcun modello e quindi calcolano ogni raccomandazione a partire dai dati grezzi. Questi ultimi impiegano un tempo notevolmente superiore a completare la richiesta di raccomandazione rispetto a quelli model-based e pertanto vengono utilizzati di rado e solo quando l'approccio con modello risulta fortemente svantaggioso.

L'approccio model-based condensa le informazioni presenti nelle matrici di interesse al fine di presentarli sotto una veste più compatta e già pronta per la raccomandazione. In particolare come modello viene spesso costruita una matrice di similarità nota come SM (Similarity Matrix) che esprime il grado di somiglianza tra diverse entità, come item o utenti. In Figura 2.3 è riportato un esempio di trasformazione della ICM. Si noti che la matrice SM è quadrata e simmetrica poiché se un item a è simile ad un item b è plausibile che l'item b sia

altrettanto simile ad a. Una volta costruita la matrice di similarità, per effettuare una raccomandazione sarà sufficiente verificare quali item tra quelli proposti sono più simili a quelli già utilizzati utilizzando una metrica adatta allo scopo.

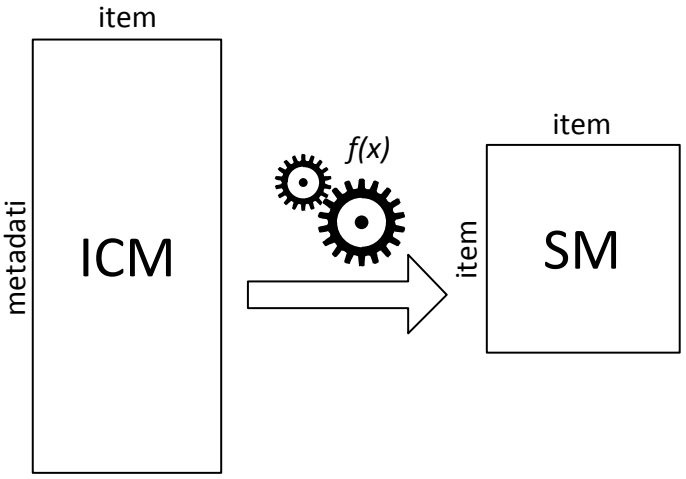


Figura 2.3: Passaggio da ICM a SM

A differenza delle matrici di partenza che sono molto sparse, la matrice SM risulta densa poiché per ogni entità deve esprimere la similarità con tutte le altre. Tenere in considerazione tutte queste informazioni in fase di raccomandazione, oltre che risultare oneroso in termini computazionali, introduce rumore dovuto ai moltissimi contributi irrilevanti e di piccola entità. É preferibile pertanto considerare solo una porzione dei dati contenente le informazioni più importanti utilizzando un algoritmo di selezione noto col nome di *k-Nearest Neighbor* o *k-NN* [18]. Per quanto riguarda la scelta del parametro k si è soliti utilizzare valori noti in letteratura, in genere pari a 200, piuttosto che optare per una vera ottimizzazione poiché risulta essere un compito eccessivamente gravoso.

Infine, quando vengono prodotti i risultati ordinati della raccomandazione, emerge come sia inutile mostrarli tutti perché l'utente non potrebbe far altro che fidarsi degli item giudicati più importanti senza la possibilità di verificarli di persona in quanto la lista completa potrebbe superare le migliaia di unità. Questo tipo di filtraggio è noto come *top-N* e mantiene i migliori N risultati proposti dalla raccomandazione. In fase di test si utilizzano valori di N compresi tra 1 e 20 così da apprezzare il comportamento dei diversi algoritmi al variare del parametro.

I metodi model-based possono essere divisi a loro volta in item-based ed user-based [46]. I primi utilizzano modelli che esprimono la somiglianza tra i diversi

oggetti mentre i secondi considerano le similitudini tra gli utenti. Ad esempio in una matrice SM user-based, con valori compresi tra 0 e 1, dove l'elemento di posto (3,7) valesse 0.9 si potrebbe concludere che gli utenti 3 e 7 hanno gusti molto simili.

In uno scenario reale in cui il numero degli utenti è molto più numeroso rispetto a quello degli elementi è più conveniente memorizzare la similarità tra item poiché lo spazio di memoria occupato cresce col quadrato della cardinalità dell'insieme. Inoltre gli algoritmi di tipo item-based sono più robusti al cambiamento poiché in caso di aggiunta di un nuovo utente o di un nuovo rating non è necessario computare nuovamente il modello perché questo ignora totalmente la presenza degli utenti. Infine anche la qualità delle raccomandazione fornite dai metodi item-based risulta essere di qualità superiore per cui spesso è la scelta dominante [57, 23].

Ogni categoria fin qui discussa è ricca al suo interno di algoritmi differenti che implementano soluzioni molto diverse tra loro. Nei paragrafi seguenti verranno analizzate più approfonditamente le singole metodologie così come gli algoritmi più rappresentativi.

2.4 Metodi non personalizzati

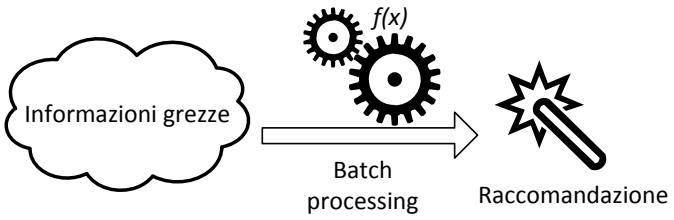


Figura 2.4: Metodi non personalizzati

Gli algoritmi non personalizzati sono tutti quegli algoritmi che non sono in grado di generare raccomandazioni ad hoc per ogni utente. Come mostrato in Figura 2.4, per elaborare la lista di raccomandazione un metodo non personalizzato condensa direttamente le informazioni grezze in quella che sarà la raccomandazione finale. Questi algoritmi sono molto semplici e vengono utilizzati quando le informazioni disponibili non sono sufficientemente numerose oppure in tutti i casi in cui ci siano delle forti restrizioni dal punto di vista della capacità di calcolo nonché dell'occupazione di memoria.

Gli algoritmi non personalizzati più utilizzati sono *Top Popular* e *Movie Average*, che sfruttano due principi molto semplici. Il primo raccomanda gli item più noti tra gli utenti mentre il secondo giudica l'importanza degli oggetti mediando le votazioni espresse dagli utenti.

2.4.1 Top Popular

L'algoritmo Top Popular [6] consiglia gli item più popolari, noti anche come blockbuster o best seller, senza considerare le valutazioni espresse dagli utenti. Questo significa che anche un film giudicato pessimo da un numero molto grande di utenti può rientrare nella lista di raccomandazione.

Per quanto l'algoritmo sembri comportarsi bene per un nuovo utente consigliandogli gli *hot items*, cioè quelli più usati, è plausibile che l'individuo li conosca già e quindi il sistema risulta essere di scarsa utilità pratica. É evidente che se il sistema di raccomandazione è implementato su una piattaforma di vendita con un magazzino molto vasto questo non farà che acuire il problema della *long tail*, ossia di quella porzione consistente di item, stimata prossima all'80%, che non diventano mai dei grandi successi [4]. In questo contesto è fondamentale che il sistema di raccomandazione riesca a valorizzare anche gli oggetti meno popolari per consentire un reale profitto.

2.4.2 Movie Average

Il metodo Movie Average [34] consiste nel calcolare la media dei rating ottenuti da ogni singolo oggetto ottenendo in questo modo una sorta di valutazione collettiva. Sebbene questa tecnica sia molto semplice consente comunque di ottenere dei discreti risultati in fase di raccomandazione poiché l'informazione che è in grado di condensare è un indice abbastanza realistico del sentimento popolare.

A differenza della soluzione Top Popular il costo computazionale è lievemente superiore poiché invece di contare il numero di valutazioni per ogni singolo item è necessario calcolare una media dei valori. Tuttavia il beneficio che si ottiene utilizzando questa strategia è tale da giustificare ampiamente l'incremento.

2.5 Metodi content-based

Gli algoritmi di tipo content-based utilizzano le informazioni relative agli oggetti per effetturare le raccomandazioni. Sfruttando l'evidenza che è molto probabile che gli individui apprezzino oggetti simili a quelli già utilizzati in passato, questi metodi consigliano quindi item somiglianti a quelli già fruiti dall'utente. La

somiglianza è basata sulla presenza o meno di caratteristiche comuni, che ad esempio per un libro possono essere il titolo, il genere, l'autore e la sinossi.

Per poter compiere raccomandazioni basate sul contenuto degli oggetti è necessario avere una profonda conoscenza degli item e delle loro caratteristiche ed anche la modalità con cui vengono memorizzate le informazioni diventa cruciale per la buona riuscita delle raccomandazioni.

L'uso di informazioni testuali, come avviene nella maggioranza dei casi, introduce una moltitudine di problemi legati alle differenze tra sintassi e semantica. Si pensi ad esempio all'autore Isaac Asimov indicato come *ASIMOV* o *I. Asimov*, oppure alle congiunzioni che potrebbero affollare la sinossi e che dal punto di vista semantico sono inutili in questo contesto. Esistono inoltre problemi dovuti ai sinonimi nonché a quei termini che possono esprimere significati distinti, come la parola *pesca*. Infine si considerino anche genere e numero di sostantivi ed aggettivi nonché le coniugazioni verbali, tutti orpelli che andrebbero rimossi per estrarre il nucleo semantico. Per ovviare a tutti questi inconvenienti è necessario svolgere un importante operazione di filtraggio prima di poter operare sui dati [50].

Esiste inoltre il problema che alcuni termini rappresentano meglio l'oggetto a cui sono legati mentre altri sono termini più comuni. E' utile pertanto assegnare pesi diversi alle varie parole dando maggiore enfasi a quelle che si ripetono spesso in un documento e che sono invece rare negli altri. Questa idea è condensata nella Formula 2.2, nota come *tf-idf*, che esprime il peso w da assegnare al termine k nel documento i [56].

$$w_{ik} = tf_{ik} \times idf_k \tag{2.2}$$

In particolare tf_{ik}, nota come *term frequency*, viene calcolata come in Formula 2.3 ed esprime l'importanza di un item in uno specifico documento come un valore compreso tra un massimo pari ad uno e un minimo pari a 0^+. Il significato di questa porzione della formula complessiva è che ai termini più ricorrenti verrà dato maggior peso mentre gli altri verrano penalizzati.

$$tf_{ik} = \frac{\#t_{ik}}{\max_i \#t_{ik}} \tag{2.3}$$

Il termine idf_k è invece la *inverse document frequency* e si ottiene dalla Formula 2.4. Lo scopo di questa parte della formula è dare maggior peso alle parole rare piuttosto che a quelle comuni assegnando valori compresi tra un massimo pari a $\log N$ e un minimo pari a zero.

$$\begin{aligned} idf_k &= \log \frac{N}{n_k} \\ n_k &= \sum_i \exists t_{ik} \\ N &= \#i \end{aligned} \tag{2.4}$$

Oltre ai problemi intriseci legati al testo, le soluzioni content-based presentano altre criticità [39] di cui la più notevole è la super-specializzazione. Ad esempio se un algoritmo utilizzasse esclusivamente l'informazione dell'autore del testo, consiglierebbe ad un utente che ha letto solo libri di Isaac Asimov testi del medesimo scrittore. In questo modo gli algoritmi content-based difficilmente riusciranno a consigliare qualche novità suggerendo semplicemente oggetti simili a quelli già visti.

Anche se la maggior parte degli algoritmi content-based lavorano su dati testuali esistono anche soluzioni che integrano con successo informazioni audio [15] e video [69].

L'indiscusso vantaggio che accomuna i metodi content-based è la capacità di poter effettuare raccomandazioni relative a nuovi oggetti di cui nessun utente ha già usufruito. La sola conoscenza dei dati riguardanti il nuovo item è sufficiente ad espandere il modello comprensivo del nuovo elemento così che si renda possibile effettuare una raccomandazione includendo il nuovo oggetto nella lista proposta.

Se è vero che tutti gli algoritmi content-based condividano i concetti fondamentali, ciascuno di essi implementa soluzioni peculiari e differenti dagli altri poiché le alternative percorribili sono innumerevoli. I metodi *Cosine Content* e *Direct Content*, ad esempio, trasformano gli item in vettori e misurano quindi la loro differenza come il coseno dell'angolo che formano. Per contro il metodo *Naïve Bayes* affronta la raccomandazione come un problema di classificazione da approcciare in modo probabilistico. *LSA Cosine* invece si affida all'estrazione di caratteristiche nascoste per catalogare i diversi item. Nei prossimi paragrafi sarà possibile analizzare nel dettaglio i singoli algoritmi per comprenderne il funzionamento e le peculiarità vincenti.

introduci metodi

2.5.1 Cosine Content

L'algoritmo Cosine Content [33] rapresenta ogni item come un vettore di modulo unitario in uno spazio che ha tante dimensioni quanti sono i metadati della ICM. Ogni dimensione può assumere valori compresi tra 0 e 1 che indicano il valore del metadato per lo specifico oggetto. Se il dataset da cui provengono i dati

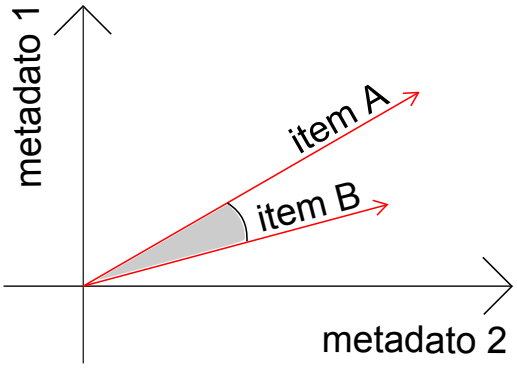


Figura 2.5: algoritmo Cosine

fosse esplicito si renderà necessaria una normalizzazione per rendere il modulo del vettore pari a 1. Se ad esempio i metadati fossero solo due, un oggetto verrebbe rappresentato in uno spazio a due dimensioni come mostrato in Figura 2.5. Riportando l'esempio in contesto letterario, se i due elementi della figura fossero libri e i metadati due generi, avventura e fantascienza, allora si potrebbe concludere che il libro A è molto più avventuroso del libro B ma riguardo la fantascienza sono molto simili.

Da un punto di vista vettoriale due oggetti appaiono identici quando i loro vettori coincidono. In questo caso infatti l'angolo individuato tra i due item è nullo e il proprio coseno è massimo e pari ad uno. Quando invece due elementi sono completamente differenti l'angolo massimo che può essere originato dai due vettori è pari a $\frac{\pi}{2}$ annullando quindi il coseno. Rapportandoci all'esempio precedente in uno spazio con due soli metadati, questo caso corrisponde alla variante in cui l'oggetto A ottenga il massimo valore per un metadato e sia nullo per il secondo e si verifichi la condizione inversa per l'altro item.

La similitudine di due item viene così rappresentata da coefficienti compresi tra zero ed uno che andranno a pesare i valori nel profilo dell'utente. In particolare, in fase di raccomandazione, ogni riga della matrice SM corrispondente a ciascun oggetto viene estratta e moltiplicata vettorialmente col profilo dell'utente, preventivamente normalizzato, per valutare nuovamente il coseno tra i due vettori. Si ottiene così che gli elementi già apprezzati dall'utente ricevano un peso maggiore mentre gli altri vengano penalizzati. Sommando infine tutti i pesi calcolati si ricava il rating per quello specifico oggetto. Ripetendo l'operazione è possibile calcolare il rating per tutti gli elementi presentandoli infine in una lista ordinata dove l'item con rating maggiore indicherà il più simile alle preferenze dell'utente.

2.5.2 Direct Content

Il metodo Direct Content è un altro metodo basato sulla rappresentazione vettoriale degli item che però, a differenza di Cosine Content non compie alcuna normalizzazione sui dati. Questo significa che le similarità calcolate e contenute nel modello saranno differenti dal precedente metodo.

Operativamente, come indicato in Formula 2.5, il modello viene generato come il prodotto tra la URM trasposta, indicata come URM^T, e la URM così da ottenere una matrice di similarità che abbia dimensioni $item \times item$. La lista di raccomandazione invece, indicata come rec_list, verrà generata come prodotto tra il profilo dell'utente e la matrice SM, come sintentizzato nelle seguenti formule.

$$\begin{aligned} model &= URM^T \times URM \\ rec\_list &= profile \times URM \end{aligned} \tag{2.5}$$

Questo algoritmo inoltre utilizza il filtraggio k-NN. Questa operazione conserva i k item più simili a quello oggetto di raccomandazione ed ignora tutti gli altri.

2.5.3 Naïve Bayes

Un classificatore bayesiano è un algoritmo che sfrutta il teorema di Bayes sulla probabilità condizionata, mostrato nella Formula 2.6, per valutare un'ipotesi A a fronte di una o più evidenze B [47]. L'assunzione che tutte le evidenze siano indipendenti tra di loro rende il calcolo più semplice e caratterizza il classificatore col termine naïve [41].

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{2.6}$$

$$\begin{aligned} P(Y = v|X_1, X_2, .., X_n) &= \frac{P(X_1, X_2, .., X_n|Y = v)P(Y = v)}{P(X_1, X_2, .., X_n)} \\ P(X_1, X_2, .., X_n) &= \sum_{j=0}^{m} P(X_1, X_2, .., X_n|Y = v_j)P(Y = v_j) \qquad m = \#Y \end{aligned} \tag{2.7}$$

$$P(X_1, X_2, .., X_n|Y = v) = \prod_{i=0}^{n} P(X_i|Y = v) \qquad \text{assunzione naïve} \tag{2.8}$$

In particolare nel problema in esame il classificatore dovrà discernere se un item possa essere o meno di interesse per un certo utente, noti i metadati dell'oggetto in esame così come di tutti quelli graditi all'utente. Dal punto di vista operativo

emergono due principali problematiche nell'implementazione del classificatore bayesiano naïve. Nel caso in cui le evidenze siano molto numerose come in Formula 2.7, il prodotto di tutte le probabilità indipendenti associate alle evidenze porterebbe ben presto all'annullamento del risultato per cui è necessario passare ad una scala logaritmica. Inoltre se la probabilità di un'evidenza è nulla l'intero prodotto, mostrato in Formula 2.8, verrebbe annullato. Questo problema può essere risolto aggiungendo ad ogni conteggio dei diversi attributi un'unità. Al crescere delle osservazioni il peso di questa approssimazione decrescerà all'incirca come $\frac{1}{x}$.

2.5.4 LSA Cosine

L'algoritmo LSA (Latent Semantic Analysis) è in grado di raggruppare dei testi in funzione dei concetti che essi esprimono [22]. Questo metodo è fondato sull'uso della decomposizione SVD di una matrice, o Singular Value Decomposition, che fattorizza una matrice di partenza $A_{m \times n}$ nella forma $A = U \times S \times V^T$ dove U e V sono matrici unitarie, cioè tali per cui $U \times U^T = U^T \times U = I$. La matrice U avrà dimensione $m \times k$ dove $k = \min(m, n)$ e ogni colonna è un autovettore della matrice $A \times A^T$. La matrice S, di dimensioni $k \times k$, è diagonale e gli elementi ivi posti, detti valori singolari, sono la radice degli autovalori della matrice $A \times A^T$ ordinati dal più grande in posizione $(1, 1)$ al più piccolo in posizione (k, k). Infine la matrice V di dimensioni $n \times k$ e ogni colonna è un autovettore della matrice $A^T \times A$. Se la matrice di partenza non ha rango massimo la decomposizione SVD è in grado di rappresentarla in una forma più compatta che ha rango massimo, come mostrato nel seguente esempio:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 1 & 1 \end{pmatrix} = U \times S \times V^T =$$

$$= \begin{pmatrix} -0.44 & 0.08 & -0.90 \\ -0.88 & 0.17 & 0.45 \\ -0.19 & -0.98 & 0 \end{pmatrix} \times \begin{pmatrix} 8.52 & 0 & 0 \\ 0 & 0.64 & 0 \\ 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} -0.28 & -0.54 & -0.80 \\ -0.87 & -0.21 & 0.45 \\ 0.41 & -0.82 & 0.41 \end{pmatrix}$$

Essendo la matrice S diagonale per costruzione, la presenza di un elemento nullo sulla diagonale consente di rimuovere dalle matrici U e V l'ultima colonna poiché risulteranno annullate nei prodotti tra matrici.

$$U \times S \times V^T = \begin{pmatrix} -0.44 & 0.08 \\ -0.88 & 0.17 \\ -0.19 & -0.98 \end{pmatrix} \times \begin{pmatrix} 8.52 & 0 \\ 0 & 0.64 \end{pmatrix} \times \begin{pmatrix} -0.28 & -0.54 & -0.80 \\ -0.87 & -0.21 & 0.45 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 1 & 1 \end{pmatrix} = A$$

Si è ottenuta così una rappresentazione più compatta della matrice di partenza A.

Se mantenendo tutti i valori della matrice S è possibile ricostruire la matrice di partenza, riducendone il rango e quindi annullando alcuni valori in ordine a partire dal più piccolo permette di costruire una matrice approssimata $\hat{A}$ che sarà tanto più simile ad A tanti più valori verranno preservati. Nell'esempio sopracitato annullando il valore più piccolo in S consente di rimuovere un'ulteriore colonna da tutte le matrici che in questa forma ridotta vengono indicate come U^T, S^T e V^T .

$$\hat{A} = \hat{U} \times \hat{S} \times \hat{V}^T = \begin{pmatrix} -0.44 \\ -0.88 \\ -0.19 \end{pmatrix} \times \begin{pmatrix} 8.52 \end{pmatrix} \times \begin{pmatrix} -0.28 & -0.54 & -0.80 \end{pmatrix} =$$

$$= \begin{pmatrix} 1.05 & 2.01 & 2.98 \\ 2.10 & 4.02 & 5.95 \\ 0.45 & 0.87 & 1.28 \end{pmatrix} \approx A$$

In questo contesto operativo il rango scelto per la matrice $\hat{S}$ è chiamato latent size e individua quante caratteristiche, o feature, verranno preservate dai dati originali. Utilizzare la matrice S originale vuol dire distinguere ogni item da tutti gli altri mentre ridurre la latent size significa accorpare semanticamente gli oggetti aumentando il livello di astrazione e quindi perdendo qualche dettaglio.

Un esempio per comprendere il funzionamento potrebbe consistere nel voler distinguere una persona in un insieme di 10 individui. Se potessimo utilizzare 10 attributi per descriverli potremmo indicare i codici fiscali di ognuno così da riuscire facilmente ad individuarli. Se il numero di aggettivi dovesse ridursi ad esempio a 4 sarebbe necessario cambiare strategia e optare per termini più generali

che coprano tutte le persone, come ad esempio il colore dei capelli che assumiamo possa essere ad esempio biondo, moro, castano o rosso. Lavorando su un numero ridotto di informazioni per ciascuna persona sarà più semplice calcolare la loro similarità ed inoltre avendoli raggruppati secondo una strategia potrebbe svelare delle similarità prima nascoste.

Si noti che in una decomposizione reale non è possibile associare ad una feature un'etichetta semantica che ne esprima il significato. Questa affermazione, riportata all'esempio precedentemente esposto, significa che troveremmo gli utenti descritti da 4 fattori latenti, ovvero nascosti, senza alcuna indicazione sul loro significato.

Il prodotto $D = \hat{S} \times \hat{V}^T$ avrà come dimensione $feature \times item$ ed esprimerà il grado di matching tra ogni oggetto e le diverse feature. Da questo è possibile ottenere il modello valutando la distanza tra gli item col metodo del coseno che, previa normalizzazione, costruisce una matrice di similarità tra item come $D^T \times D$. La raccomandazione sarà quindi ottenuta dal prodotto tra il profilo utente e il modello.

In Figura 2.6 è possibile osservare un esempio dell'effetto della decomposizione SVD e successiva ricostruzione su un'immagine, dove le dimensioni della matrice S originale sono state ridotte di cinquanta volte. Si noti come le caratteristiche complessive più macroscopiche siano rimaste rendendo l'immagine comunque riconoscibile.

2.6 Metodi collaborativi

I metodi collaborativi utilizzano le informazioni relative alla fruizione degli item da parte degli utenti per effettuare le raccomandazioni [30]. Come nella vita reale è possibile affidarsi ai consigli degli amici così il sistema di raccomandazione si fonda sul concetto di similarità, o vicinanza, tra gli utenti per produrre le raccomandazioni sfruttando due congetture: persone simili avranno in media la stessa considerazione degli item e oggetti simili otterranno le stesse valutazioni dai diversi utenti. Basandosi su questi due assiomi è possibile, in un contesto collaborativo, stabilire una somiglianza tra i diversi item in funzione del gradimento espresso dagli utenti e quindi raccomandare ad un utente gli oggetti più simili a quelli per cui ha già espresso un parere positivo.

Poiché gli algoritmi collaborativi non hanno bisogno di analizzare le caratteristiche degli item, sono completamente immuni ai problemi riscontrati nei metodi content-based dovuti all'analisi dei contenuti così come la specializzazione. Offrono inoltre il grande vantaggio di poter raccomandare all'utente un oggetto

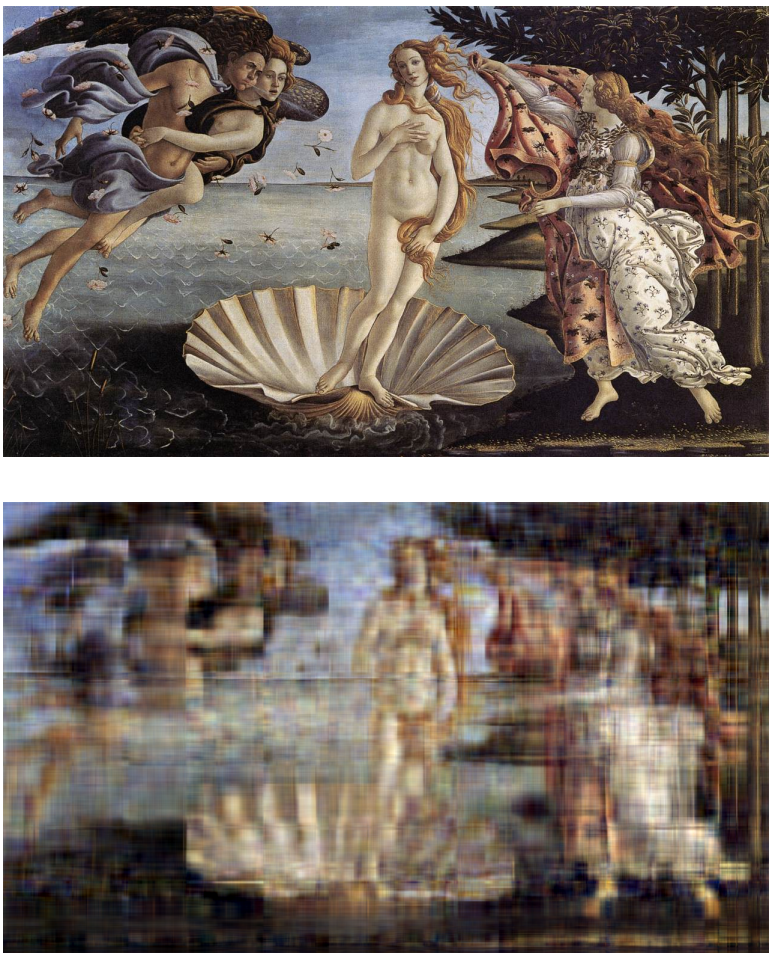


Figura 2.6: Immagini originale e compressa con SVD a confronto

inatteso che non assomigli a quelli già utilizzati dall'utente ma che tuttavia è stato giudicato interessante dagli altri utenti con gusti simili. Lo stato di stupore nel trovare qualcosa di bello e inaspettato mentre si stava cercando altro, noto col nome di *serendipity* [64], è il principale punto di forza dell'approccio collaborativo.

Per contro gli algoritmi appartenenti a questa famiglia sono proni ad altri problemi di cui il più rilevante è l'impossibilità di raccomandare un item finché un numero sufficiente di individui non abbia espresso un giudizio su di esso [59]. Infatti per come sono ideati questi metodi se nessun utente ha lasciato un parere riguardo ad un oggetto questo risulterà non avere oggetti simili e pertanto non verrà mai presentato in alcuna raccomandazione. Inoltre poiché il funzionamento del sistema è legato alla presenza delle valutazioni è necessario che queste siano presenti in numero sufficiente perché le raccomandazioni siano significative e questo rappresenta un punto critico quando la matrice URM risulta essere molto sparsa come nei casi reali [1]. Una possibile soluzione per alleviare quest'ultimo problema è ricorrere all'uso di URM implicite che spesso sono più popolate delle rispettive esplicite poiché vengono costruite in maniera trasparente all'utente e quindi senza la necessità di alcuna interazione.

In caso venissero raccolte valutazioni esplicite ci si potrebbe imbattere in ulteriori problematiche dovute alle modalità con cui votano gli utenti. Si noti infatti che in media i best seller, per loro natura, avranno molti più rating rispetto agli altri oggetti e questo porterà il sistema di raccomandazione a prediligerli rispetto agli altri item. L'Esempio 2.9 mostra questo fenomeno dove il terzo item, i cui rating sono riportati nella terza colonna della matrice URM, è stato utilizzato da quasi tutti gli utenti. La creazione di un modello che esprima la similarità tra gli oggetti, utilizzando un algoritmo basato sul coseno e descritto nel Paragrafo 2.6, evidenzia come il terzo item risulti il più simile a tutti gli altri, escluso il quinto. I valori evidenziati mostrano come in caso di raccomandazione verrà prediletto il terzo oggetto su tutti gli altri non tanto per la sua qualità quanto per la notorietà, nascondendo potenziali item più interessanti per l'utente.

$$URM = \begin{pmatrix} 0 & 0 & 4 & 2 & 1 \\ 1 & 4 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 1 & 3 & 1 & 4 \\ 0 & 4 & 3 & 0 & 0 \\ 2 & 0 & 4 & 0 & 0 \end{pmatrix} \tag{2.9}$$

$$model = Cosine(URM)$$

$$model = \begin{pmatrix} 1 & & & & \\ 0.28 & 1 & & & \\ \mathbf{0.47} & \mathbf{0.43} & \mathbf{1} & & \\ 0 & 0.07 & \mathbf{0.65} & 1 & \\ 0 & 0.15 & \mathbf{0.56} & 0.63 & 1 \end{pmatrix}$$

In generale si osserva inoltre che i best seller hanno una valutazione media più elevata rispetto agli altri item, poiché graditi da un folto numero di utenti, e che taluni individui sono soliti assegnare valutazioni più elevate della media mentre altri si dimostrano invece più critici. Per ovviare a questi problemi si rende necessaria una fase iniziale di filtraggio che consenta di ridurre il rumore dei dati così da ottenere una qualità superiore in fase di raccomandazione [7].

Un tipo di normalizzazione utile per risolvere questo problema e nota come *baseline estimates* si propone di rimuovere questi effetti considerandoli come la somma di tre componenti distinte, ognuna che rappresenta un particolare fenomeno [34].

$$b_{ui} = \mu + b_u + b_i \tag{2.10}$$

Come mostrato in Formula 2.10 la baseline b_{ui} stimata per l'utente u e l'item i è composta dalla media μ di tutti i rating e dalle baseline b_u e b_i. La baseline b_u è calcolata per ogni utente u come differenza tra la media di tutti i rating che ha espresso e la media μ complessiva. In maniera analoga la baseline b_i per ogni

item i è calcolata come la differenza tra la media dei rating che ha ottenuti dagli utenti e la media μ.

Una volta calcolato per ogni rating il contributo dovuto alla baseline è possibile quindi rimuoverlo per rivelare la parte di informazione più importante e significativa.

2.6.1 Cosine k-NN

Il metodo Cosine k-NN, come l'analogo metodo Cosine Content per i metodi content-based presentato nel Paragrafo 2.5.1, costruisce il modello dei dati calcolando la somiglianza tra due item come il coseno dei vettori che li rappresentano. L'uso del filtraggio k-NN, discusso nel Paragrafo 2.3, consente inoltre di migliorare la qualità delle raccomandazioni rimuovendo dal modello i contributi meno rilevanti.

In fase di raccomandazione, similmente al metodo content-based, ogni riga del modello viene moltiplicata vettorialmente col profilo dell'utente al fine di misurare la distanza tra il vettore che esprime i gusti dell'individuo e quello che indica la similarità tra l'item estratto e tutti gli altri.

2.6.2 Pearson k-NN

L'algoritmo Pearson k-NN [60] utilizza il coefficiente di correlazione di Pearson per valutare la somiglianza tra due diversi item. Come mostrato nella Formula 2.11 seguente il coefficiente ρ_{xy}, che indica la similarità tra gli item x ed y, è dato dal rapporto tra la covarianza σ_{xy}, tra x e y, e il prodotto delle due deviazioni standard σ_x e σ_y. Si noti che X ed Y indicano i profili degli item x ed y mentre T indica il profilo di un generico oggetto t.

$$\begin{aligned} &\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \cdot \sigma_y} \\ &-1 \leq \rho_{xy} \geq 1 \\ \\ &\sigma_{xy} = E[(X - E(Y))(Y - E(X))] \\ &\sigma_t = \sqrt{\sigma_t^2} \\ &\sigma_t^2 = E[(T - E(T))^2] \end{aligned} \tag{2.11}$$

Dall'analisi del coefficiente di correlazione è possibile evincere quanto siano simili gli item tra loro. In particolare se $\rho_{xy} > 0$ allora le due variabili sono

correlate positivamente, se $\rho_{xy} < 0$ sono correlate negativamente mentre per $\rho_{xy} = 0$ risultano non correlate.

In fase di raccomandazione il profilo normalizzato dell'utente viene moltiplicato vettorialmente col profilo di ciascun item così da produrre una lista che indichi la similarità tra i gusti dell'individuo e gli item. Questa lista di raccomandazione verrà infine ordinata e presentata all'utente.

2.6.3 PureSVD

Il metodo PureSVD [19] è il corrispettivo collaborativo del metodo content LSA Cosine discusso nel Paragrafo 2.5.

Sfruttando la decomposizione matriciale SVD è possibile ottenere una nuova rappresentazione del dataset che rappresenti per ogni item la sua vicinanza alle feature selezionate. A partire da questa matrice intermedia D viene calcolata la matrice di similarità SM come il prodotto $D' \times D$. Questa modalità di misurare la somiglianza tra oggetti assomiglia ad un coseno fatto salvo per la mancanza di normalizzazioni. Partendo dalla URM che ha dimensioni $feature \times item$ si passa quindi allo spazio dei fattori latenti D che ha dimensioni $metadati \times item$ e da qui alla matrice di similarità $item \times item$.

2.6.4 AsymmetricSVD

L'algoritmo AsymmetricSVD nasce dall'unione di diverse tecniche di raccomandazione con lo scopo di sfruttare i singoli punti di forza di ciascuna [34]. La tipologia di algoritmi che fonde sono *item-based neighborhood*, cioè metodi che valutano la similarità degli item, e *latent factor models*, ovvero algoritmi che fanno uso della decomposizione SVD.

In particolare questo metodo prevede una fase di ottimizzazione che minimizza l'errore quadratico medio dei rating con una discesa del gradiente, producendo come conseguenza valori di fallout molto bassi. Tra tutti i metodi presentati AsymmetricSVD è certamente quello che dipende dal maggior numero di parametri e ciò comporta un'importante fase di tuning preliminare dalla quale dipende la qualità finale delle raccomandazioni generate.

Da un punto di vista formale i rating vengono calcolati come segue.

$$\hat{r}_{ui} = b_{ui} + q_i^\intercal \left(|R(u)|^{-\frac{1}{2}} \cdot \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot x_j \right) \tag{2.12}$$

Dove $\mathbf{x}_j$ e $\mathbf{q}_i^\intercal$ sono i vettori dei fattori latenti che vengono precedentemente calcolati minimizzando una funzione di errore usando la discesa del gradiente.

$$\min_{\mathbf{q},\mathbf{x},\mathbf{b}} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - \mathbf{q}_i^\intercal \cdot \left(|R(u)|^{-\frac{1}{2}} \cdot \sum_{jinR(u)} (r_{uj} - b_{uj}) \cdot \mathbf{x}_j \right) \right)^2 +$$
$$+ \lambda \cdot \left(b_u^2 + b_i^2 + \|\mathbf{q}_i^\intercal\|^2 + \sum_{j\in R(u)} \|\mathbf{x}_j\|^2 \right)$$

Dove $\mathcal{K}$ indica un insieme di coppie <user, item> estratte casualmente che sono usate per calcolare l'errore tra il rating reale e quello calcolato dall'algoritmo. Si noti che il secondo addendo della formula ha il solo scopo di favorire l'uso di valori ridotti per i parametri penalizzando di fatto i valori più elevati.

2.7 Metodi ibridi

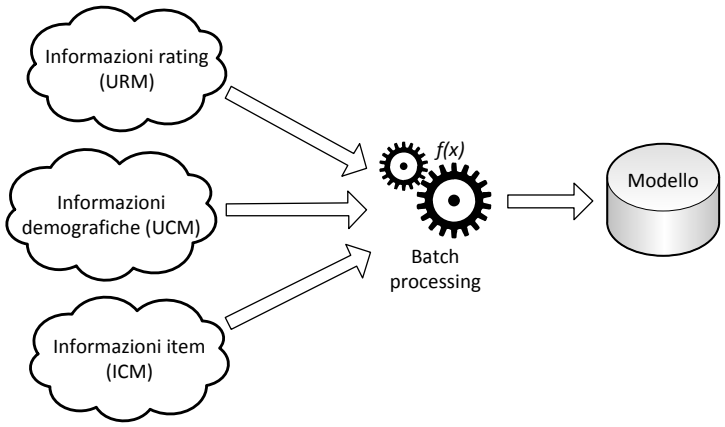


Figura 2.7: Metodi ibridi

In generale i metodi ibridi sono tutti quegli algoritmi che non utilizzano solo una tipologia di informazioni, come URM o ICM, ma le accorpano con strategie differenti al fine di ottenere raccomandazioni che includano diversi aspetti dello stesso fenomeno [10]. In Figura 2.7 è riportata ad esempio la famiglia dei metodi ibridi che fanno uso delle informazioni contenute nelle matrici URM, ICM ed UCM. All'interno di una famiglia coesistono algoritmi differenti che condividono i dati ma che implementano soluzioni differenti per giungere alla raccomandazione.

L'idea che ha portato alla nascita dei sistemi ibridi è che unendo tipi eterogenei di metodi è possibile sfruttare le peculiarità di ciascuno di essi con l'obiettivo di migliorare le performance e di lenire i punti di debolezza dei singoli algoritmi. Questa fusione però non porta automaticamente ad un miglioramento poiché

ad un aumento delle informazioni disponibili non sempre segue un incremento significativo della bontà dei risultati, essendo questi influenzati dagli specifici algoritmi utilizzati nonché dall'introduzione di possibile rumore.

In base alla tecnica utilizzata per unire i contributi dei singoli metodi sono state identificate un vasto numero di famiglie di algoritmi [11]. É infatti possibile, ad esempio, produrre diverse liste di raccomandazione distinte con ciascun metodo per poi combinarle prima di presentarle all'utente seguendo una determinata strategia, come implementato dagli algoritmi *mixed*, *weighted* e *switching*. Un'altra alternativa presente in letteratura, nota come *feature combination* [62], consiste nel produrre direttamente una sola lista di raccomandazione utilizzando tutti i dati a disposizione. Ulteriori algoritmi si propongono di effettuare raccomandazioni utilizzando i diversi metodi uno dopo l'altro così che l'uscita del primo costituisca l'ingresso per il secondo, come nella famiglia *cascade*. Anche la produzione del modello utilizzando un algoritmo e la generazione delle raccomandazioni con uno differente è contemplata tra le alternative, col nome di *meta-level*. Esistono infine metodi che utilizzano i dati content per aumentare la qualità dei dati collaborativi, sui quali verranno calcolate le raccomandazioni [3].

Si discuteranno nei seguenti paragrafi le caratteristiche di ciascun metodo con particolare enfasi all'algoritmo Hydra, discusso nel Paragrafo 2.7, scelto per i test condotti in questo lavoro di tesi poiché in grado di sfruttare tutte le informazioni fornite dai diversi dataset.

2.7.1 Mixed

Questa tipologia di algoritmi ibridi combina i risultati ottenuti da sistemi diversi, ad esempio collaborativo e content [61], secondo una determinata politica come ad esempio presentare prima le raccomandazioni ottenute da un metodo e poi le altre oppure alternare i risultati provenienti dai diversi algoritmi.

Per quanto un algoritmo di questo tipo sia molto semplice da realizzare la sua qualità è fortemente connessa alle qualità dei diversi sistemi di raccomandazione utilizzati e la presenza di anche un solo metodo inefficace porta al degrado della bontà delle raccomandazioni.

2.7.2 Weighted

I metodi ibridi weighted consistono nel comporre la raccomandazione finale pesando i risultati di ciascun metodo con un coefficiente che ne indica l'importanza [25]. Da un punto di vista formale ogni rating $r_u^n i$, prodotto dall'algoritmo n

per l'utente u relativamente all'item i, viene pesato dal coefficiente α_n proprio di ciascun metodo come mostrato in Formula 2.13.

Anche questa soluzione come la precedente è facilmente realizzabile ma la qualità è fortemente dipendente dai pesi utilizzati.

$$\begin{aligned} r_{ui} &= \alpha_1 \cdot r_{ui}^I + \alpha_2 \cdot r_{ui}^{II} + \cdots + (1 - \sum_i^{N-1} \alpha_i) \cdot r_{ui}^N \\ \sum_i^N \alpha_i &= 1 \end{aligned} \tag{2.13}$$

2.7.3 Switching

Questa famiglia di metodi deriva da quella *mixed* dove il rating scelto per un particolare item è quello fornito dall'algoritmo di cui si ha maggior confidenza [9]. Un semplice esempio potrebbe essere l'utilizzo dei rating ottenuti da un algoritmo content-based per gli item con pochi rating mentre sarebbe plausibile l'uso dei rating provenienti da un metodo collaborativo per gli altri casi.

Per quanto l'idea di scegliere per ogni item l'algoritmo che meglio lo descriva, effettuare una stima verosimile della confidenza potrebbe risultare un problema di difficile soluzione.

2.7.4 Feature combination

Gli algoritmi appartenenti a questa famiglia combinano informazioni eterogenee, provenienti ad esempio da URM ed ICM, prima di calcolare il modello e quindi le raccomandazioni. Un metodo appartenente a questa famiglia é l'algoritmo Hydra, trattato nella prossima sezione.

Il vantaggio principale di questa soluzione è che viene compiuta una sola raccomandazione invece che una per ogni metodo utilizzato, diminuendo notevolmente il carico computazionale. Sorge però il problema sulla normalizzazione dei dati poiché, provenendo da sorgenti differenti, potrebbero avere caratteristiche molto diverse non sempre confrontabili. Ad esempio se è sempre possibile rimappare un range di valori su un altro, non è univoca la scelta da compiere in caso di diverse densità dei dati.

Hydra

L'algoritmo Hydra [62] fa uso delle informazioni contenute nelle matrici URM, ICM ed UCM. In particolare le matrici vengono composte come mostrato in Figura 2.8 originando una nuova matrice detta Extended Rating Matrix che ha

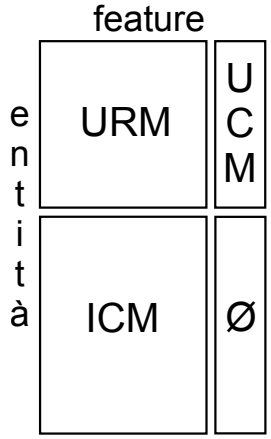


Figura 2.8: Composizione della Extended Rating Matrix nell'algoritmo Hydra

per righe delle entità, composte da utenti e metadati, e per colonne le feature delle entità, formate da item e informazioni demografiche relative agli utenti. L'area che si origina nella porzione in basso a destra ed indicata con $\emptyset$ risulta vuota poiché non c'è alcun tipo di relazione tra gli item e le caratteristiche proprie solo degli utenti. Prima dell'assemblaggio della ERM le matrici che la costituiscono vengono normalizzate al fine di rendere la media dei valori diversi da zero pari a uno. Della matrice così ottenuta viene calcolata la decomposizione SVD ottenendo un modello che esprime la vicinanza tra un sottoinsieme delle feature e le entità. Per compiere infine una singola raccomandazione è necessaria solo la porzione del modello relativa agli utenti mentre la restante parte può essere eliminata.

2.7.5 Cascade

I metodi di questo gruppo utilizzano diversi algoritmi in cascata in modo che ad ogni passo vengano selezionati parte degli item di origine, con una struttura a livelli profonda a piacere [10]. Il primo algoritmo selezionerà così i *top-n* item secondo la propria metrica e passerà i risultati al secondo che effettuerà la raccomandazione considerando solo gli item selezionati al primo passo.

Per questi metodi risulta critica la scelta dell'ordine degli algoritmi poiché in generale vale la relazione mostrata in Formula 2.14, dove f e g rappresentano i due diversi sistemi di raccomandazione, I è l'insieme degli item e X è l'altra dimensione della matrice utilizzata come sorgente dati, ad esempio le feature oppure gli utenti.

$$
\begin{aligned}
& f(g(I,X),X) \neq g(f(I,X),X) \\
& f : I \times X \to I \\
& g : I \times X \to I
\end{aligned}
\tag{2.14}
$$

2.7.6 Meta-level

Gli algoritmi che fanno parte di questa famiglia utilizzano un metodo per produrre il modello ed uno appartenente ad una classe differente per compiere le raccomandazioni. Ad esempio si potrebbe costruire un modello partendo da informazioni content per poi produrre le raccomandazioni utilizzando metodi della classe collaborativa [49].

Questo approccio consente di ottenere un risparmio in termini di tempo rispetto alle altre soluzioni poiché il modello è calcolato con un solo step invece che passando per i diversi algoritmi ma la qualità ottenuta dipende come nel caso *Cascade* dalla scelta dell'ordinamento degli algoritmi da eseguire.

2.7.7 Feature augmentation

Questa famiglia di algoritmi introduce delle nuove informazioni in un dataset ricavandole da un altro. Ad esempio è possibile stimare dei rating per un utente utilizzando un classificatore bayesiano naïve sui dati content, come proposto nell'algoritmo CBCF (Content Boosted Collaborative Filtering) [42]. La probabilità che un utente gradisca un certo item conoscendo la similarità tra tutti gli item può essere usata per popolare la matrice URM. A questo punto può essere usato un algoritmo collaborativo per effettuare le raccomandazioni.

Lo svantaggio principale di questo metodo è che tutto il processo viene svolto a *runtime* poiché è necessario sapere per quale utente si sta effettuando la stima dei rating e questo può risultare molto oneroso dal punto di vista computazionale.

Questo specifico problema è stato superato utilizzando nuovi metodi come l'algoritmo FFA che sono in grado di separare la creazione del modello dalla raccomandazione [3]. Come mostrato in Figura 2.9 i dati content vengono utilizzati nello stadio CBF per creare un modello e compiere una raccomandazione per ogni utente. Dopo uno stadio di filtraggio che elimina i contributi meno significativi i dati vengono combinati con quelli collaborativi generando una URM aumentata, o aURM. A questo punto è possibile procedere con la creazione di un modello utilizzando algoritmi collaborativi.

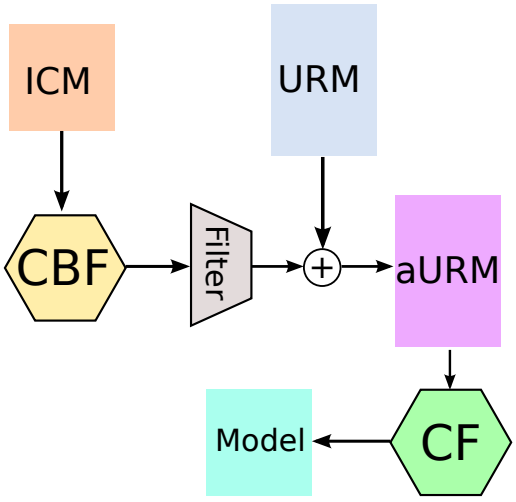


Figura 2.9: Schema di funzionamento dell'algoritmo FFA tratto dal paper originale [3].

Capitolo 3

Analisi del problema

In questo capitolo viene analizzato il problema di compiere raccomandazioni in presenza di un canale di distribuzione unidirezionale. A partire dal Paragrafo 3.2 vengono proposti e discussi svariati approcci al problema, sfruttando diverse famiglie di sistemi di raccomandazione. Viene inoltre compiuta, nel Paragrafo 3.7, un'analisi relativa ai requisiti hardware per l'implementazione delle soluzioni. Infine il Paragrafo 3.8 affronta la tematica della privacy nel campo dei sistemi di raccomandazione.

3.1 Introduzione

Scopo di questa tesi è valutare il comportamento dei sistemi di raccomandazione in uno scenario di content delivery su canale di comunicazione unidirezionale, come ad esempio le piattaforme digitale terrestre e satellitare.

Come mostrato in Figura 3.1 lo scenario è composto da due tipi di entità: il service provider e la televisione con set-top box, ossia fornita di un dispositivo atto a permetterle di ricevere un particolare segnale televisivo. Per il tipo di relazione che si instaura tra il provider e il set-top box, dove il primo eroga un servizio e il secondo ne fruisce, i due attori dello scenario vengono anche indicati rispettivamente come server e client.

Il service provider identifica un'azienda che si occupa di trasmette contenuti televisivi, o item, verso gli utenti che dispongono di una televisione con set-top box, come un decoder per il digitale terrestre. Il fornitore del servizio ha inoltre l'esigenza di suggerire ai propri clienti contenuti che possano risultare di loro gradimento, per spingerli all'acquisto in un ottica di pay-per-view (PPV) o per aumentare la *customer satisfaction*. Sul mercato esistono una moltitudine di fornitori di contenuti che erogano servizi in maniera gratuita o a pagamento ed il

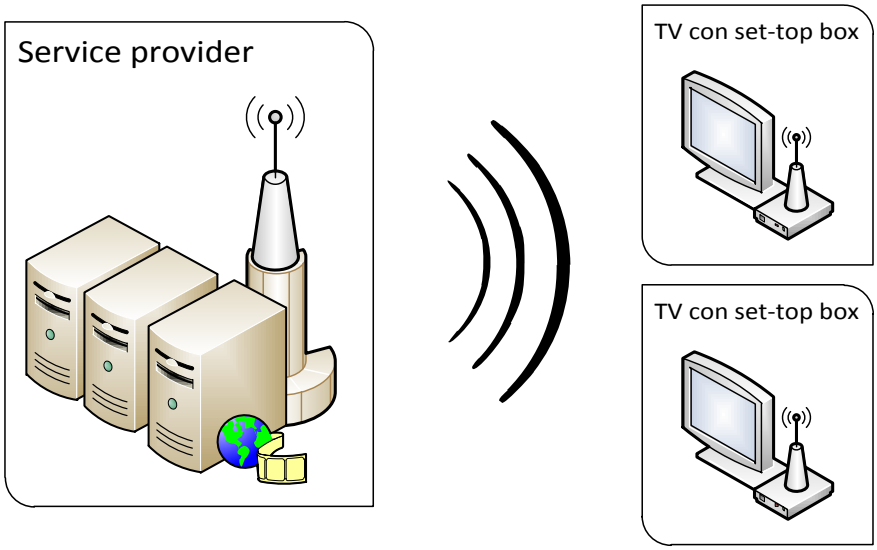


Figura 3.1: Scenario di trasmissione su canale broadcast unidirezionale

medesimo service provider può adottare entrambe le soluzioni per diversificare la propria offerta. I ragionamenti che seguiranno sono applicabili indistintamente ad entrambe le tipologie di fornitore.

Il service provider e il set-top box possono essere connessi attraverso diverse tipologie di canale di comunicazione che, dal punto di vista del flusso delle informazioni, possono risultare bidirezionali o unidirezionali. Un canale bidirezionale è un canale che consente al set-top box di inviare informazioni al service provider mentre un canale unidirezionale preclude questa possibilità. Esempi d'uso di canali bidirezionali si possono osservare nella IPTV, che sfrutta l'infrastruttura internet, oppure nella pay-per-view (PPV) poiché prevede un meccanismo che permetta all'utente di interagire col fornitore per l'acquisto di un contenuto televisivo. Fanno invece uso di canali unidirezionali le emittenti del digitale terrestre o l'offerta satellitare, quando prive di contenuti PPV. Questi collegamenti non consento in alcun modo a due set-top box differenti di poter comunicare poiché realizzano un tipo di connessione broadcast dove il service provider trasmette il medesimo segnale verso tutti i destinatari contemporaneamente.

Un sistema di raccomandazione generico ha bisogno di conoscere le preferenze di un utente per potergli suggerire delle informazioni personalizzate. Se la presenza di un canale bidirezionale consente al service provider di ricevere dati dagli utenti, un canale unidirezionale nega di fatto qualsiasi flusso di informazioni dal set-top

box impedendo la produzione di raccomandazione. La diffusione capillare di quest'ultimo tipo di canale rende però necessaria un'analisi del problema più approfondita per valutare se la limitazione fisica del canale è superabile, obiettivo che questa tesi si propone di raggiungere.

A tale scopo verranno affrontate ed ampiamente analizzate diverse metodologie appartenenti a ciascuna delle famiglie dei sistemi di raccomandazione: non personalizzati, content-based, collaborativi ed infine ibridi. La scelta più naturale in un ambiente così definito è quella di optare per sistemi di raccomandazione non personalizzati poiché non necessitano *by design* del canale di ritorno ignorando totalmente le preferenze degli utenti e suggerendo a tutti i medesimi contenuti. Una soluzione content-based potrebbe facilmente costruirsi un modello dei dati poiché sono dipendenti dagli item proposti e non dagli utenti, ma resterebbe il problema di compiere la raccomandazione per ogni singolo set-top box. I sistemi collaborativi, poiché interamente basati sulle preferenze degli utenti, si scontrano con il limite fisico del canale che impedisce al service provider di collezionare qualsiasi sorta di informazione. Infine i metodi ibridi, poiché sono costituiti unendo strategie differenti, soffriranno di tutte le limitazioni dei metodi che andranno a sfruttare.

Nei paragrafi seguenti vengono discusse più in dettaglio le singole famiglie di sistemi di raccomandazione al fine di carpirne le potenzialità e i punti di debolezza in questo nuovo contesto.

3.2 Approccio non personalizzato

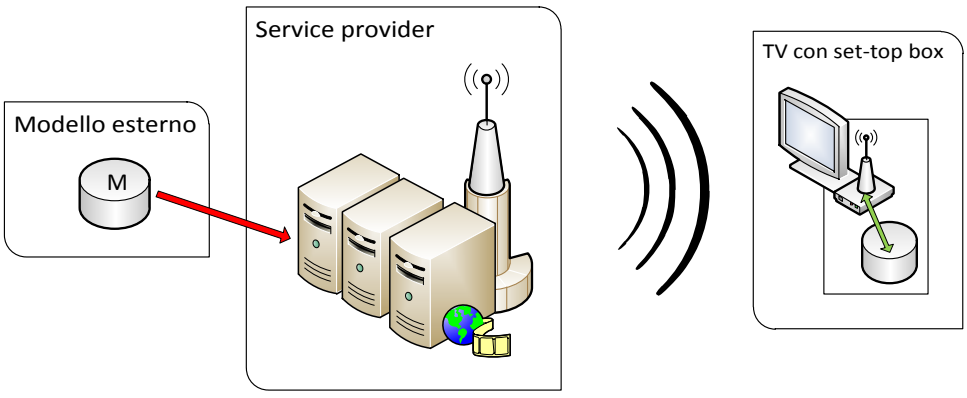


Figura 3.2: Soluzione non personalizzata

Il caso più semplice di raccomandazione è quella non personalizzata, ovvero uguale per tutti gli utenti indistintamente, come mostrato in Figura 3.2. Poiché non è possibile desumere le informazioni statistiche degli utenti del sistema a causa del canale unidirezionale, queste devono essere importate in altro modo. Si potrebbe ad esempio ricorrere a strategie di mercato precostituite così come far affidamento su dati collezionati da indagini di mercato o da altri service provider che dispongono di canali bidirezionali.

Il primo caso potrebbe ad esempio fornire le raccomandazioni in funzione dei contenuti trasmessi, per cui al termine di un programma televisivo sportivo potrebbe essere proposta la visione di un altro contenuto rigurdante lo sport, oppure consigliare item in base all'orario, suggerendo ad esempio contenuti di cucina per l'ora di pranzo.

L'uso invece di informazioni statistiche raccolte su un canale differente presenta, anche se in maniera rilassata, un certo legame con i dati reali. Conoscere, ad esempio, la tipologia di individui che guardano la televisione alle ore 17 consente di proporre raccomandazioni senza doversi basare su mere congetture come nel primo caso. Questo approccio, giudicato più razionale, verrà utilizzato nelle prove compiute in questo lavoro di tesi.

3.3 Approccio demografico

Un'evoluzione dell'approccio non personalizzato è quello demografico, che impiega informazioni legate alle caratteristiche dell'utente, quali ad esempio età e genere, per rendere le raccomandazioni personalizzate. In questo contesto il fornitore del servizio potrebbe stabilire una classe di individui, ad esempio donne dai 30 ai 40 anni, a cui suggerire uno specifico contenuto.

La mancanza di un canale di distribuzione bidirezionale, che consenta alle informazioni di transitare dall'utente al provider, impone di separare il processo di raccomandazione tra server e client. Il primo si occuperà quindi di stabilire quali contenuti sono adeguati per ciascuna categoria di utenti, costruendo delle liste di raccomandazione personalizzate per ciascuna di esse che verranno inviate ai client unitamente ai contenuti. Per contro il set-top box avrà il compito di memorizzare le caratteristiche demografiche dell'utente così da poter scegliere quale lista di raccomandazione ricevuta dal provider è più adeguata.

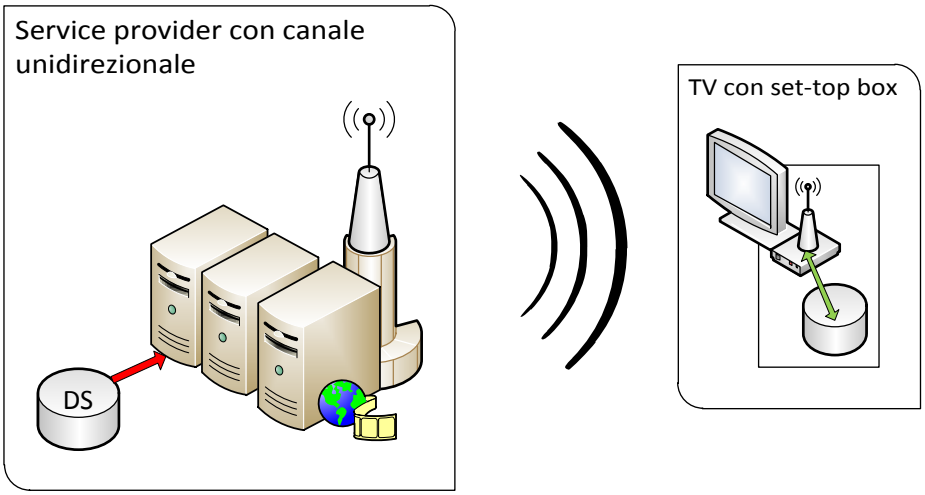


Figura 3.3: Soluzione content-based

3.4 Approccio content-based

Questo tipo di approccio si basa sulle informazioni relative ai contenuti, come ad esempio genere e durata, che pertanto consentono la costruzione di un modello dei dati poiché facilmente reperibili. La raccomandazione vera e propria, poiché dipende dalle preferenze del singolo utente, deve essere generata da ciascun set-top box per produrre risultati personalizzati.

L'approccio content-based può essere affrontato in due modi sostanzialmente differenti tra loro che chiameremo *client-side* e *server-side*.

Soluzione client-side

Per sviluppare questa soluzione content-based, esemplificata in Figura 3.3, il fornitore del servizio deve memorizzare le informazioni, o feature, relative ad ogni contenuto per poi inviarle ai set-top box, come ad esempio titolo, genere e anno.

Lato client, ovvero sul set-top box del cliente, verranno invece memorizzati tanti contatori quante sono le feature osservate così da tener traccia dei termini più interessanti per l'utente, come nell'esempio mostrato in Tabella 3.1. L'informazione memorizzata potrebbe indicare se un item con quella specifica feature sia stato visto o meno così come l'eventuale giudizio esplicito fornito dall'utente.

Nel primo caso verrà acquisita in modo trasparente per l'utente l'informazione sulla fruizione dei diversi item.

Nel secondo caso invece il giudizio sarà rilevato chiedendo all'utente di esprimere una valutazione sui contenuti visti mediande una scelta multipla di opzioni, come ad esempio un indice numerico tra 1 e 5 oppure una scelta tra possibili aggettivi come pessimo, scarso, sufficiente, buono, ottimo. Per questa soluzione è necessario memorizzare anche il numero di occorrenze di ciascuna feature così da poter correttamente calcolare il nuovo indice di gradimento qualora capitasse di valutare una caratteristica già presente nel sistema. Ad esempio ,utilizzando una semplice media pesata, se fosse presente il genere action con votazione 5 e due occorrenze e un nuovo contenuto del genere action fosse valutato 2 allora la nuova votazione da assegnare al genere action sarebbe $\frac{5 \cdot 2 + 2 \cdot 1}{2+1} = 4$ con numero di occorrenze pari a 3.

Quando il service provider vorrà proporre un nuovo contenuto invierà a tutti gli utenti le informazioni relative al contenuto e sarà quindi il set-top box di ciascun utente a valutare se il contenuto può essere di interesse per l'utente e quindi è opportuno consigliarlo.

genere	**gradimento**
action	3
adventure	5
animation	1
unknown	1

Tabella 3.1: esempio di memorizzazione dei generi

Questa soluzione idealmente applicabile presenta tuttavia delle criticità una volta portata nel contesto reale di utilizzo a causa dell'elevato costo computazionale nonché dello spazio di memorizzazione richiesto. L'uso intensivo di risorse di calcolo è dovuto al fatto che l'intera computazione viene svolta sul set-top box del cliente. Dal punto di vista della memorizzazione invece per ogni singolo oggetto sarà necessario immagazzinare in locale un gran numero di contatori che probabilmente non verranno mai più utilizzati. Si ricordi a tal proposito che le informazioni relative agli item potrebbero essere estratte anche dalla sinossi del film così da farne crescere in maniera esplosiva il numero in brevissimo tempo.

Il tentativo di ridurre i requisiti, mantenendo ad esempio solo un numero ridotto di informazioni, potrebbe condurre ad un peggioramento della qualità delle raccomandazione poiché queste sarebbero fondate su dati che descrivono il fenomeno senza troppi dettagli. É auspicabile pertanto trovare una soluzione che

riesca almeno in parte a spostare i requisiti verso il service provider che potrebbe disporre facilmente di spazio di memorizzazione e risorse computazionali.

Soluzione server-side

Invece di inviare tutte le informazioni relative ad un contenuto per effettuare la computazione lato client, è possibile spostare lato server la parte più onerosa del calcolo inviando ai set-top box un'informazione condensata che sia sufficiente per effettuare la singola raccomandazione.

Utilizzando questo approccio il service provider si occuperà di calcolare la similarità tra i contenuti di cui dispone per poi inviarli all'utente. Le informazioni relative ad ogni contenuto sono facilmente accessibili per i fornitori di servizi per cui è lecito assumere che essi dispongano di informazioni corrette e complete su tutti gli item che hanno in catalogo.

In modo analogo al caso client-side il set-top box si occuperà di memorizzare informazioni che indichino la visione del contenuto oppure una valutazione esplicita fornita dall'utente. La differenza è che in questo approccio si rende necessario mantenere una sola informazione per ogni contenuto visto mentre nel caso client-side vengono memorizzate per ogni item tante variabili quante sono le feature dell'item. Nel caso di giudizi espliciti sarà sempre necessario memorizzare l'informazione addizionale riguardante il numero di occorrenze.

Nel'istante in cui il fornitore del servizio decidesse di promuovere un contenuto dovrebbe inviare a tutti i propri clienti informazioni riguardo la similarità dell'item prescelto con tutti gli altri presenti in catalogo. Il set-top box, che custodisce il profilo dell'utente con l'elenco dei contenuti visti e il relativo punteggio, si occuperà di valutare se il nuovo item può essere di interesse per l'utente e nel caso si prenderà carico di effettuare il suggerimento. La scelta di inviare il contenuto corredato dalla lista di similarità con gli altri item garantisce lato client minori requisiti di memoria poiché una volta ricevuti i dati e calcolate le raccomandazioni non è necessario mantenere memorizzate le similarità. Inoltre non importa che il set-top box sia sempre in funzione, come richiesto in caso di invii asincroni di contenuti e liste di similarità, perché nel momento in cui il client scopre un nuovo contenuto troverà corredata la relativa lista di similarità.

3.5 Approccio collaborativo

La presenza di un canale unidirezionale impedisce totalmente la creazione di un modello poiché l'insieme dei dati collaborativi, contenente le valutazione degli utenti relative ai contenuti da essi fruiti, è fisicamente distribuito in tutti i set-top

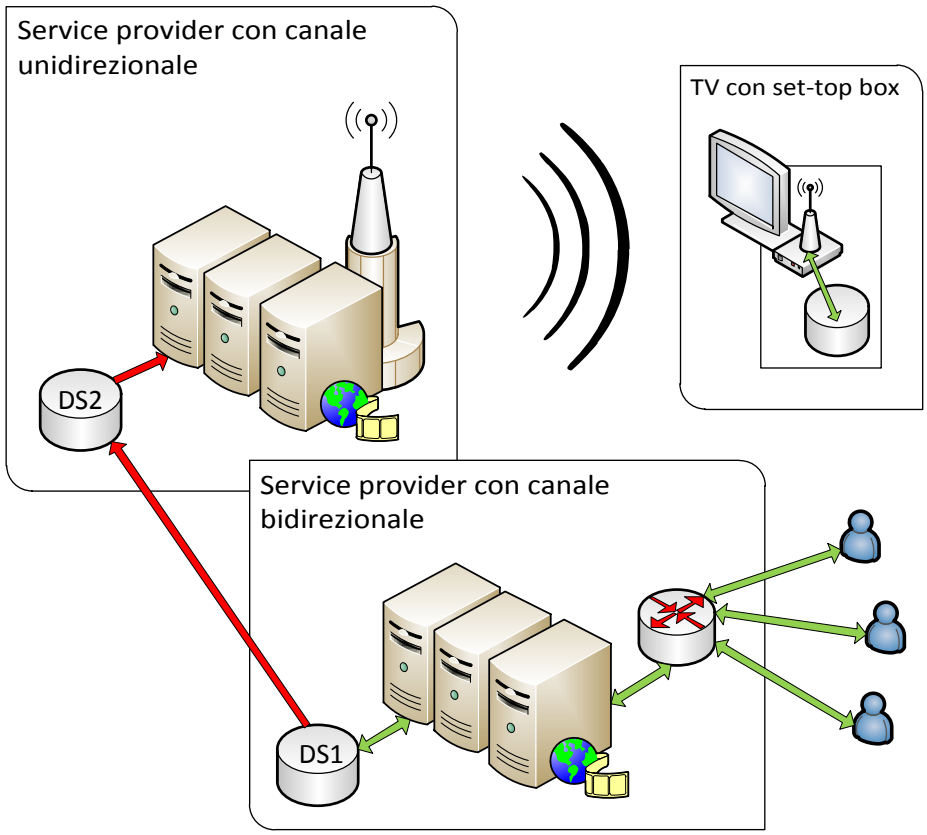


Figura 3.4: Soluzione collaborativa

box è non vi è modo di aggregarlo. Per risolvere il problema in questo lavoro di tesi si propone l'idea innovativa di utilizzare un modello dei dati appreso su un dominio diverso da quello dove poi verranno compiute le raccomandazioni. Il dominio di utilizzo del modello, dove è necessario calcolare le raccomandazioni, viene definito dominio target mentre quello utilizzato per apprendere il modello è detto dominio ausiliario.

Il modello ottenuto con questa strategia può essere impiegato come nel caso content-based per indicare al client la similarità tra il contenuto proposto e tutti gli altri item presenti in catalogo. Sarà compito del set-top box quello di memorizzare informazioni sui contenuti già visionati, in maniera implicita o esplicita, così come di valutare se un nuovo contenuto è potenzialmente di interesse per il cliente.

Questa metodologia può essere applicata qualora il service provider sia parte di una realtà più grande che eroga altri servizi televisivi su canali bidirezionali oppure un diverso service provider, sempre operante su un canale bidirezionale, decida di cedere parte delle informazioni che ha acquisito. In entrambi i casi il fornitore del servizio importerà un modello collaborativo precalcolato sul dominio ausiliario indicante la somiglianza tra i diversi item di cui dispone.

Nello scenario presentato in Figura 3.4 è mostrato come un service provider con canale bidirezionale possa apprendere un modello acquisendo i dati dei propri clienti e sia quindi in grado di produrre raccomandazioni personalizzate. Se il modello così appreso viene ceduto ad un diverso fornitore, questa volta con canale unidirezionale, si rende possibile la produzione di raccomandazioni locali sui diversi set-top box ma di tipo collaborativo.

3.6 Approccio ibrido

Con approccio ibrido si intende una metodologia che unisca due o più tecniche di raccomandazione eterogenee, come ad esempio collaborativa e content-based. Il concetto alla base di questa strategia è che fondendo approcci differenti è possibile sfruttare un dataset più ampio ed eterogeneo con l'oppurtunità di creare modelli potenzialmente più efficaci. Si noti che l'uso combinato di approcci differenti introduce tutti i vincoli dei singoli metodi, come ad esempio la memorizzazione dei contatori all'interno dei set-top box.

In questo specifico contesto è possibile operare con due diverse modalità di approccio ibrido, che indicheremo come *incrementale* e *completa*.

Modalità incrementale

Si consideri ad esempio un caso in cui si renda necessario compiere una raccomandazione collaborativa riguardante un item di cui non si disponga delle informazioni necessarie. In questo caso è possibile appoggiarsi ad una tecnica differente, ad esempio content-based, per sopperire ai dati mancanti.

Questa modalità prevede di costruire un modello primario da utilizzare come prima scelta nella produzione di raccomandazioni e di predisporre anche un modello secondario, costruito usando un diverso approccio, per supplire alle eventuali mancanze del primo. I due differenti approcci utilizzati dovrebbero essere selezionati in maniera che il primo garantisca la migliore qualità in fase di raccomandazione mentre il secondo sia in grado di coprire tutto l'insieme di lavoro a scapito ovviamente della bontà delle raccomandazioni.

L'uso di questa modalità ha il vantaggio di poter essere introdotta quando il sistema di raccomandazione è già stato realizzato con una singola metodologia poiché non necessita di alcuna modifica ma solo di un'estensione completamente indipendente dal resto del sistema. Per contro i dati disponibili vengono utilizzati solo in parte e questo potrebbe ridurre la bontà delle raccomandazioni.

Modalità completa

Una modalità differente consiste invece nell'usare un approccio completamente ibrido, cioè che faccia uso di metodi e dati eterogenei per compiere ogni singola raccomandazione. In questo caso il modello verrà calcolato utilizzando tutti i dati a disposizione e indicherà come per gli altri approcci una somiglianza tra i diversi contenuti disponibili.

Questa soluzione è da preferirsi a quella incrementale poiché è potenzialmente in grado di migliorare la qualità di tutte le raccomandazioni avendo a disposizione una quantità superiore di dati sulla quale operare. Per questo motivo la modalità completa è stata utilizzata nelle successive fasi di analisi e testing.

3.7 Requisiti hardware

La necessità di creare le raccomandazioni lato client introduce un problema di requisiti hardware per il set-top box poiché deve disporre di sufficiente capacità di calcolo così come di storage. In particolare sarà necessaria una memoria permanente dove memorizzare il profilo utente contenente le sue preferenze, una memoria volatile sulla quale operare ed una unità di elaborazione in grado di svolgere le operazioni richieste.

Dal punto di vista computazionale una raccomandazione deve essere calcolata in un tempo ridotto che possiamo imporre pari a 0.5 secondi. Ogni raccomandazione prodotta per i nuovi item richiederà in un primo tempo di mappare il profilo utente col profilo del contenuto per poi calcolarne l'affinità. Saranno pertanto richiesti tanti confronti quanti sono gli elementi nel profilo più lungo per poi calcolare tante somiglianze quanti sono gli elementi del profilo più breve. Assumendo ad esempio che entrambi i profili contengano 1000 elementi saranno necessarie 2000 operazioni per singolo contenuto. Ipotizzando di voler effettuare una raccomandazione tra 100 item differenti si produrranno 200k istruzioni che, nella condizione pessima di 10 cicli di clock per istruzione senza pipeline, richiedono una frequenza di clock di almeno 2MHz per essere completate in 0.5 secondi. La stima fornita è largamente approssimativa poiché non è dato conoscere l'architettura delle unità di calcolo reali impiegate nei set top-box così come sono ignoti i ritardi introdotti dalle diverse gerarchie di memoria. Inoltre la possibile presenza di altro software in esecuzione sulla medesima unità di calcolo rende la stima difficoltosa.

Per quanto concerne la memorizzazione lato client, assumendo sempre un profilo di 1000 elementi con l'aggiunta di altrettanti indici per il mapping sui profili dei contenuti, saranno necessari almeno 2000 spazi di memoria. Se per ogni elemento venisse utilizzato un comune intero, rappresentato con 4 byte, il sistema dovrebbe disporre di almeno 8kB di memoria permanente.

La memoria volatile non ha invece particolari requisiti di spazio poiché gli elementi possono essere caricati di volta in volta dalla memoria permanente. É indubbio che una quantità di memoria pari ad almeno quella stimata di 8kB porterà benefici in termini di tempo di esecuzione della raccomandazione poiché verranno eliminati gli overhead dovuti alle operazioni di trasferimento tra memorie.

Al fine di comprendere se i dispositivi reali possano soddisfare i requisiti fin qui descritti, in Tabella 3.2 vengono mostrati alcuni esempi di set-top box scelti tra quelli disponibili sul mercato corredati dalle caratteristiche tecniche più salienti.

In particolare la memoria volatile è indicata come SDRAM mentre quella permanente corrisponde a Flash e HDD. É segnalata inoltre la presenza di una porta USB che consente il collegamento con un'unità di memorizzazione permanente esterna.

Analizzando requisiti e caratteristiche tecniche è possibile concludere che anche se le stime elaborate sono lontane dall'essere affidabili per carenza di informazioni tecniche più specifiche, le dimensioni delle grandezze dei supporti reali sono tali da far supporre con una buona confidenza che il nuovo approccio alle raccomandazioni presentato in questo lavoro di tesi possa essere già implementato sull'hardware

attualmente disponibile.

3.8 Privacy

I sistemi di raccomandazione, per loro natura, collezionano informazioni relative agli utenti. Per quanto lo scopo finale di memorizzare i dati è quello di produrre raccomandazioni di valore e quindi utili per l'utente, i sistemi di raccomandazione rappresentano una seria minaccia alla privacy degli individui [36]. Oltre al fatto che il gestore del sistema dispone di tutte le informazioni personali degli utenti, esiste il rischio concreto che parte dei dati possano essere estratti mediante attacchi mirati.

In letteratura sono riportate diverse tipologie di attacchi volti ad estrarre il profilo degli utenti. Ad esempio un utente malintenzionato potrebbe effettuare inferenze statistiche sugli utenti creando nuovi profili fittizi, con valori di rating inseriti ad hoc, per valutare i cambiamenti delle raccomandazioni [51]. Altre tecniche passive sfruttano invece i cambiamenti nell'esito delle raccomandazioni, che avvengono con l'evoluzione dei profili degli individui nel tempo, per inferire informazioni sugli utenti del sistema [12].

L'utilizzo di sistemi di raccomandazione su canali di distribuzione unidirezionali modifica significativamente il livello della privacy degli utenti rispetto ai sistemi tradizionali. Infatti l'impossibilità per il set-top box di inviare le informazioni al service provider garantisce un completo anonimato, sia nei confronti del gestore del servizio che rispetto alle tecniche di attacco proposte. In particolare la prima tipologia di attacco risulterebbe vana poiché non è possibile inserire alcun profilo nel sistema. La seconda invece consentirebbe di inferire informazioni ma sul dominio ausiliario e non su quello target dove risiedono i clienti. Anche nell'eventualità di porting del modello, le informazioni trasferite da un provider ad uno differente sarebbero del tutto prive di ulteriori informazioni utili ad identificare gli utenti.

Da questa analisi si può concludere che i sistemi di raccomandazione implementati su canali di distribuzione unidirezionali preservano integralmente la privacy dei propri utenti, garantendo al contempo la fornitura di buone raccomandazioni personalizzate.

Canale	Offerta commerciale	Hardware	Frequenza CPU	SDRAM	Flash	HDD	USB
Satellitare	MySky HD	Samsung DSB-P990N		256MB	128MB	320GB	
Digitale Terrestre	Mediaset Premium NetTV	Tele System TS7900HD	400MHz	256MB	32MB	no	si
Via Cavo	Verizon FiOS TV	Motorola 7100-P2 HD	400MHz	256MB	32MB	no	si
Digitale Terrestre		i-CAN 1100T	200MHz	64MB	16MB	no	no
Ethernet	Alice TV	Pirelli STB HY101	300MHz	128MB	64MB	no	si

Tabella 3.2: Esempi di STB con relative caratteristiche tecniche. I valori mancanti non sono reperibili.

Capitolo 4

Metodologia proposta

In questo capitolo viene presentata la metodologia impiegata per questo lavoro di tesi. Inizialmente è stata compiuta una fase preliminare di test su dataset artificiali, come discusso nel Paragrafo 4.1, per valutare la possibilità di effettuare il porting del modello. A seguire sono stati svolti nuovi test su dataset reali, riportati nel Paragrafo 4.2, per osservare il comportamento del porting tra domini realmente disgiunti. Quest'ultima fase di test ha richiesto l'ottimizzazione degli algoritmi, presentata nel Paragrafo 4.3, così da garantire la miglior performance di ciascun metodo.

4.1 Test su dataset artificiali

Al fine di comprendere se l'idea alla base di questo lavoro di tesi fosse realmente applicabile, cioè il porting di un modello su un dataset differente da quello di apprendimento, il lavoro di testing su due dataset reali distinti è stato preceduto dall'esecuzione di prove su dataset artificiali al fine di ottenere una valutazione preventiva.

Gli algoritmi selezionati per questa fase di test sono della famiglia dei collaborativi perché sono gli unici per i quali sia sensato effettuare il porting. Infatti per i metodi content è plausibile che tutti i service provider dispongano delle informazioni relative ai contenuti per cui i modelli generati sarebbero i medesimi. Inoltre i parametri che governano il funzionamento dei diversi metodi sono stati imposti così come proposto nei relativi paper che li descrivono senza alcuna ottimizzazione. La scelta, in questa fase preliminare, di condurre test utilizzando algoritmi non ottimizzati è stata dettata dalla volontà di ottenere solo risultati qualitativi per comprendere le linee generali che regolano il fenomeno oggetto di studio. Nei test seguenti, discussi a partire dal Paragrafo 4.2, verranno utilizzati un numero

maggiore di algoritmi e ciascuno verrà ottimizzato per trarne la massima qualità di raccomandazione.

Per generare dataset artificiali che potessero però contenere dati reali è stato scelto come sorgente il dataset Netflix, presentato nel Paragrafo 5.2. L'insieme dei dati è stato così partizionato opportunamente in due nuovi dataset utilizzando due strategie differenti.

Il primo test condotto ha visto la URM divisa in training set e test set in modo che non vi fossero gli stessi utenti presenti in entrambi gli insiemi, come mostrato in Figura 4.1. Il training set rappresenta il dominio ausiliario, per il quale sono disponibili le informazioni, mentre il test set è utilizzato come dominio target, ovvero dove sono incogniti i dati e si vuole compiere le raccomandazioni. Sono stati compiuti quattro test facendo variare la dimensione del training set posto di volta in volta pari a 0.01%, 0.1%, 1% e 10% dell'URM dove 0.01% corrisponde a una porzione di circa 50 utenti. Il test set invece è stato ottenuto per differenza insiemistica tra l'URM è il training set.

Questi valori sono stati scelti per verificare la qualità del porting di modelli appresi su dataset più piccoli di quelli dove verranno compiute le raccomandazione. Si noti che per valori superiori al 10% non sono state riscontrate variazioni significative delle caratteristiche misurate.

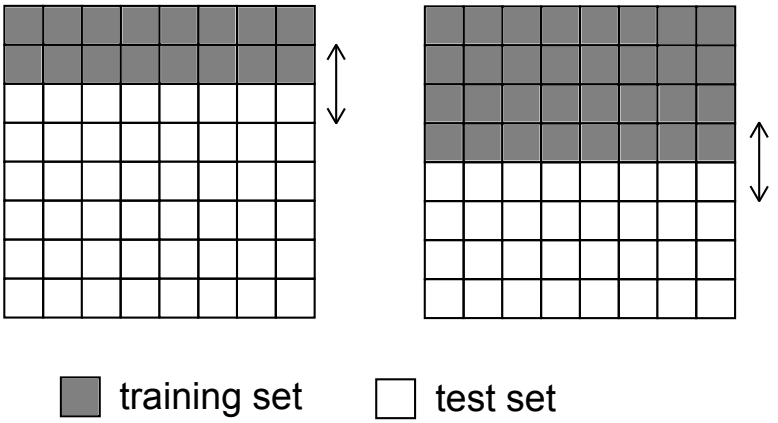


Figura 4.1: Prima variante del partizionamento del dataset Netflix

Il secondo test è stato svolto sempre su training set e test set disgiunti, a rappresentare i domini ausiliario e target, ma al posto di variarne la dimensione è stata alterata la densità del training set, come rappresentato in Figura 4.2. Il test set è costituito da metà della URM originale di Netflix mentre dall'altra metà sono stati estratti parte dei rating per lo svolgimento dei test. I valori di densità scelti per effettuare le prove sono pari a 0.01%, 0.1%, 1% e 10% della metà del dataset dove 0.01% corrisponde a circa 5000 rating.

Lo scopo di questo test è quello di valutare come cambia la qualità delle raccomandazioni se il dataset da cui viene appreso il modello contiene poche informazioni. Come nel caso precedente anche in questo caso non si sono verificate variazioni importanti per valori oltre il 10%.

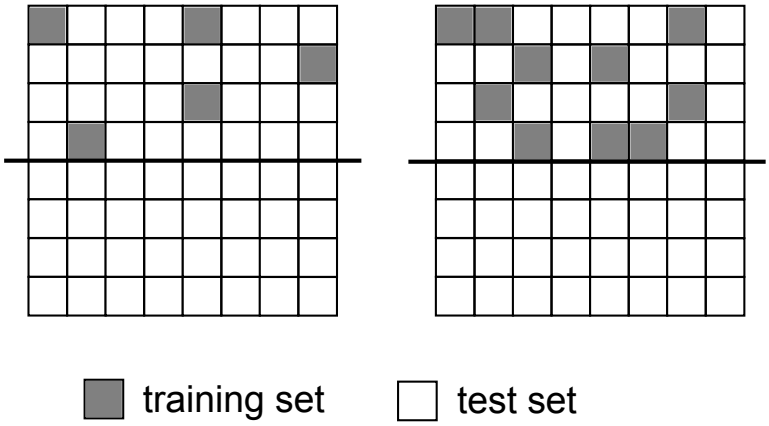


Figura 4.2: Seconda variante del partizionamento del dataset Netflix

4.2 Test su dataset reali

Una volta avuta la conferma sulla bontà dell'idea di portare il modello da un dataset ad uno differente, si è potuto procedere con i test su insiemi di dati reali. Per lo scopo sono stati utilizzati i dataset MovieLens e Yahoo!, presentati nel Paragrafo 5.1, per la loro peculiarità di condividere parte degli item.

Per gli algoritmi collaborativi il modello è stato appreso e testato su ogni dataset così da realizzare tutte le possibili combinazioni, come mostrato in Tabella 4.1.

caso	**training set**	**test set**
1	MovieLens	Yahoo!
2	MovieLens	MovieLens
3	Yahoo!	MovieLens
4	Yahoo!	Yahoo!

Tabella 4.1: Casi testati sui dataset MovieLens e Yahoo! per gli algoritmi collaborativi

Questa scelta ha permesso di confrontare le raccomandazioni ottenute nel caso di porting con quelle ricavate nel caso ideale in cui il dataset sia noto. Inoltre è stato possibile verificare come i due insiemi di dati, con caratteristiche differenti, si comportino in presenza del porting del modello.

Con la stessa modalità riassunta in Tabella 4.1 sono stati testati anche gli algoritmi non personalizzati. Si noti che per questo genere di algoritmi non ha senso parlare di modello poiché l'esito della raccomandazione non è influenzato dal profilo dell'utente, per cui si è effettuato un vero e proprio porting dei risultati ottenuti da un dataset ad uno differente.

Si noti invece che l'insieme dei test appena presentato non sarebbe stato significativo per gli algoritmi content poiché per un service provider è sempre possibile acquisire queste informazioni. Per questo motivo il modello è stato appreso da un dataset condiviso, considerato il migliore disponibile (*best*), ed è stato in seguito testato sui due diversi dataset, come sintetizzato in Tabella 4.2.

Lo scopo di questi test è stato quello di valutare la differenza tra metodi content-based e collaborativi in questo nuovo contesto dove il canale di distribuzione è unidirezionale.

caso	**training set**	**test set**
a	*best*	Yahoo!
b	*best*	MovieLens

Tabella 4.2: Casi testati sui dataset MovieLens e Yahoo! per gli algoritmi content-based

Poiché i dataset utilizzati per effettuare i test contengono anche informazioni demografiche è stato possibile valutare anche il comportamento di un metodo ibrido che fosse in grado di sfruttare tutte le informazioni apportate da URM, ICM e UCM.

Al fine di ottenere da ciascun algoritmo la massima qualità in questi test su dataset reali si è resa necessaria un'importante fase di ottimizzazione dei singoli metodi, descritta approfonditamente nei paragrafi seguenti.

4.3 Algoritmi

Gli algoritmi presenti in letteratura, ampiamente trattati a partire dal Paragrafo 2.4, sono metodi pensati e realizzati assumendo sempre la presenza del canale di ritorno. Si è resa quindi necessaria un'opera di revisione dei vari metodi così da

renderli compatibili col nuovo ambiente operativo. La mancanza in letteratura di metodi demografici non ha permesso di effettuare test relativi a questa famiglia. Come discusso nel Paragrafo 3.1, tutti i metodi eccetto quelli content-based sono stati testati nel caso in cui i dati relativi ad utenti ed item siano sconosciuti e perciò sia stato necessario importarli da un model provider, cioè qualunque altro service provider che decida di condividere le proprie informazioni.

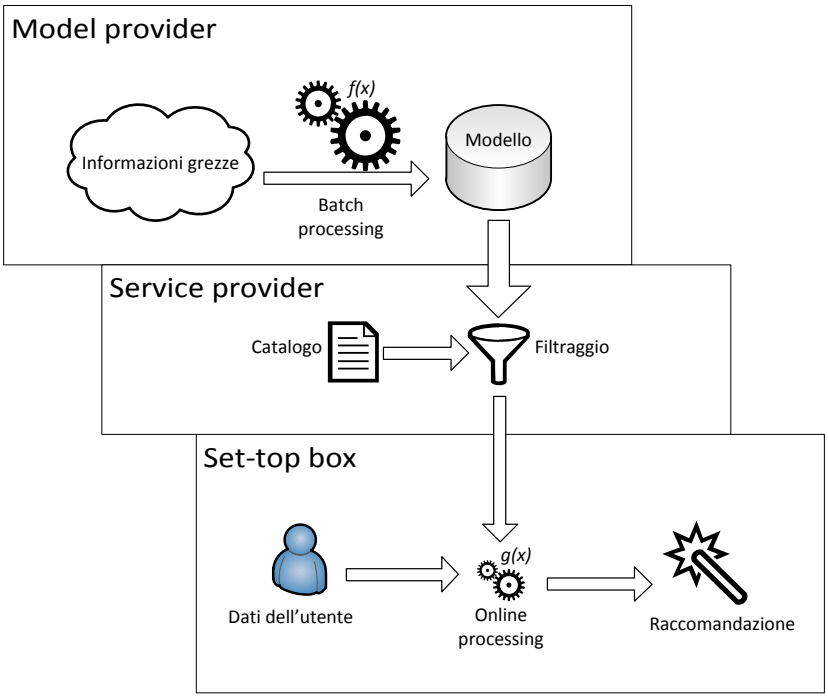


Figura 4.3: Raccomandazioni in assenza di un modello proprietario

Per molti algoritmi si è quindi configurato il caso in cui il meccanismo di raccomandazione si trovi fisicamente separato tra macchine differenti. Se il service provider non è in grado di generare alcun modello, questo compito verrà demandato ad un apposito partner lasciando al fornitore del servizio la possibilità di selezionare solo le informazioni utili. Sarà il set-top box presso l'utente finale a compiere la raccomandazione sfruttando le informazioni del cliente stesso e quelle inviate dal service provider. Uno schema del funzionamento è mostrato in Figura 4.3.

Di seguito vengono analizzati approfonditamente i singoli metodi raccolti per famiglie di algoritmi: non personalizzati, content-based, collaborativi e ibridi.

4.3.1 Metodi non personalizzati

I metodi non personalizzati, pur essendo i più semplici proposti, non sono implementabili in un contesto client-side, cioè dove la computazione venga effettuata lato client. Questo accade perché hanno bisogno di informazioni riguardanti gli altri utenti e in questo contesto non è possibile disporne. La raccomandazione pertanto può basarsi o su regole definite ad hoc o su rilevazioni effettuate su altri dataset.

Si potrebbe ad esempio procedere utilizzando delle direttive empiriche come:

$$\begin{aligned} if &\quad \text{orario}.isFasciaProtetta() \\ then &\quad \text{pubblicità}.selectOnlySafe() \\ else &\quad \text{pubblicità}.selectAll() \end{aligned}$$

$$\begin{aligned} if &\quad \text{programmaInCorso}.getArgomento() == \text{cucina} \\ then &\quad \text{pubblicità}.selectGenere(\text{femminile}) \end{aligned}$$

Sebbene questo genere di strategia venga correntemente utilizzata nel panorama pubblicitario televisivo, è evidente come sia completamente indipendente dagli utenti e quindi scorrelata dai dati per cui l'intero modello è affidato semplicemente a delle supposizioni.

Nei test svolti in questo lavoro di tesi si è invece proceduto con il produrre le raccomandazioni su un dataset differente per poi testarle su quello interessato. Si noti che non vi è propriamente il porting del modello quanto il vero e proprio uso dei risultati finali ottenuti su un dataset differente, poiché i metodi non personalizzati non sono in alcun modo influenzati dal profilo dell'utente per il quale la raccomandazione viene calcolata.

4.3.2 Metodi content-based

I metodi content-based, per come sono ideati, non necessitano del canale di ritorno poiché non devono apprendere nulla dalla scelta degli utenti. Soffermandosi sulle informazioni relative agli item oggetto della raccomandazione è per tanto possibile utilizzare gli algoritmi senza alcuna modifica significativa.

La scelta di estrarre dai dataset due diverse ICM, come mostrato nel Paragrafo 5.1, ha consentito di preferire la soluzione migliore per ogni singolo algoritmo testato così da non influenzare la qualità con una scelta a priori. É così emerso che ad eccezione dell'algoritmo *LsaCosine* tutti gli altri hanno raggiunto qualità superiori utilizzando la versione della ICM senza informazioni provenienti dalle sinossi

e dalle recensioni. Questa evidenza dimostra come l'aumento di dati disponibili non sia sempre direttamente responsabile della qualità delle raccomandazioni e che questo dipenda strettamente dalle caratteristiche dei dataset nonché dagli algoritmi utilizzati.

Vengono di seguito riportati tutti gli algoritmi per i quali si è resa necessaria una fase preliminare di tuning dei parametri, così da assicurare i risultati migliori.

Direct Content

Il metodo Direct Content, presentato nel Paragrafo 2.5.2, è basato sul calcolo della similarità tra gli oggetti con il metodo del coseno. L'aggiunta del filtraggio k-NN rimuove le similarità meno forti tra gli item mantenendo per ciascun item i k item più simili. Questa scelta è giustificata dal tentativi di rimuovere il possibile rumore, introdotto dalle somiglianze più deboli, che potrebbe sporcare le raccomandazioni.

La scelta del parametro risulta pertanto vitale per la qualità delle raccomandazioni che ha come limite inferiore la soluzione Cosine Content, versione semplificata basata sempre sul coseno ma senza alcun tipo di filtraggio. I valori testati in questo lavoro di tesi, pari a 50, 100 e 200, sono stati collezionati dai paper presenti in letteratura e testati sui due diversi dataset, come mostrato in Figura 4.4.

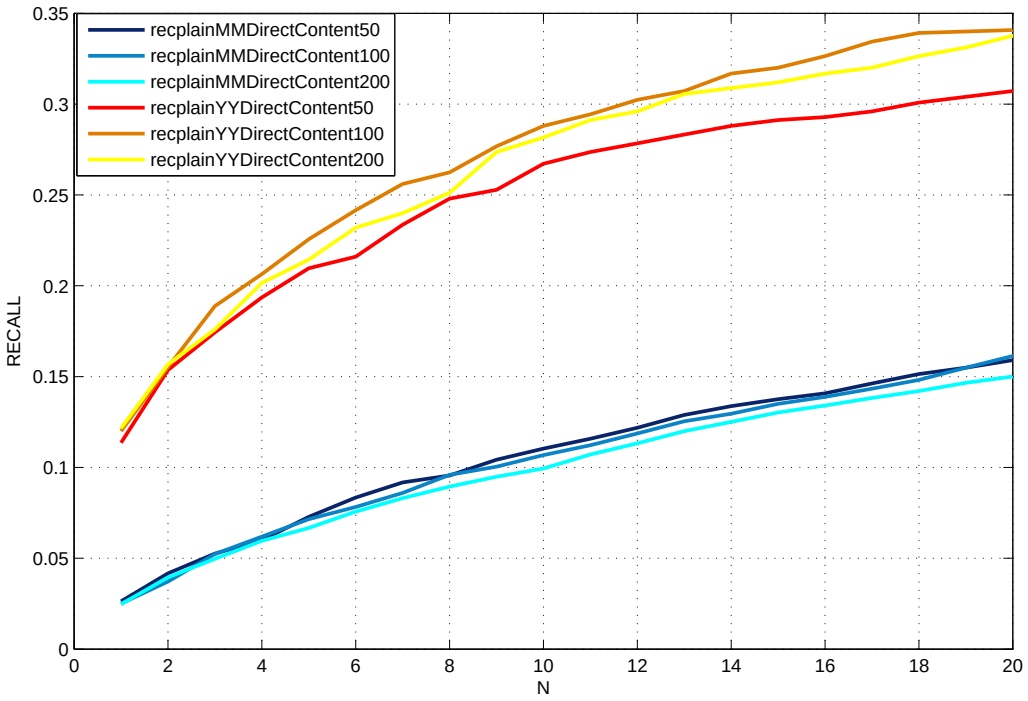


Figura 4.4: Curve della recall per diversi valori di k per l'algoritmo Direct Content

Dal grafico emerge chiaramente come il valore migliore del parametro k sia pari a 100 per entrambi i dataset, con maggior rilevanza per Yahoo!. Il grafico riporta solamente le curve relative all'uso della ICM base, priva dei dati aggiuntivi di sinossi e recensione, poiché questa versione risulta dare una qualità superiore nelle raccomandazione come si evince dal grafico in Figura 4.5. In particolare per il dataset di Yahoo! la differenza è molto marcata mentre MovieLens presenta una bontà generalmente inferiore a Yahoo! senza alcuna differenza per l'uso delle due diverse ICM.

Essendo la ICM comune ed estranea ai due dataset i modelli che ne derivano sono identici, pertanto si può concludere che la differenza così marcata tra i risultati debba essere ricercata all'interno delle rispettive URM dei dataset, che contengono i profili degli utenti utilizzati nella fase di raccomandazione.

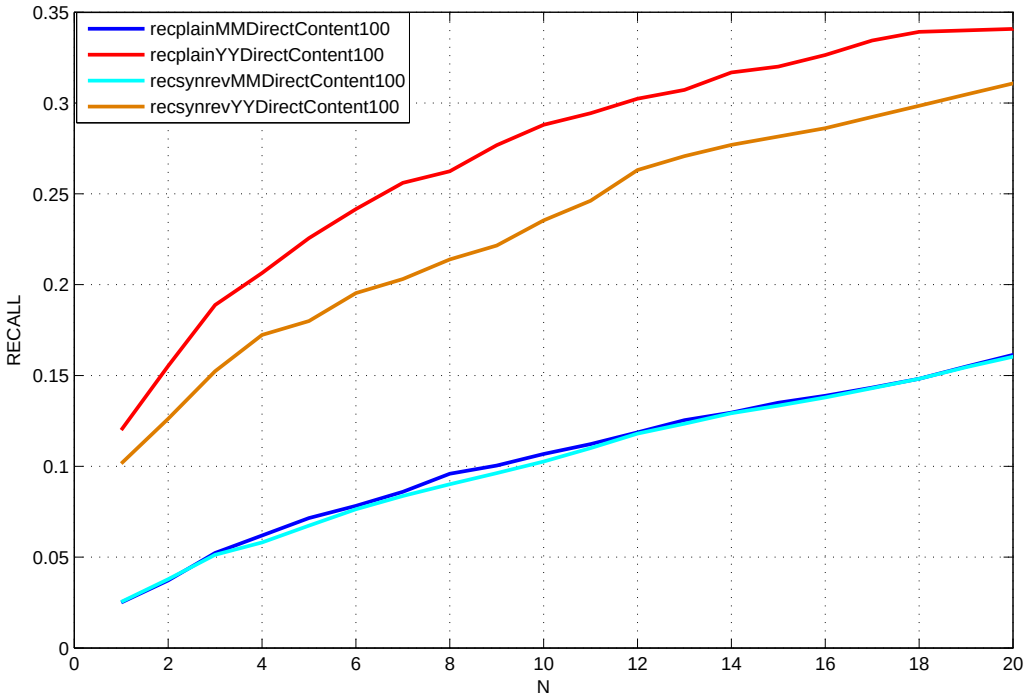


Figura 4.5: Curve della recall al variare della ICM per l'algoritmo Direct Content

LsaCosine

L'algoritmo LsaCosine è un algoritmo content-based che fa uso dellla decomposizione SVD per estrarre le feature nascoste relative agli item, come ampiamente discusso nel Paragrafo 2.5.2. Se mantenendo un numero di fattori latenti pari all'originale consente di ricreare il dataset originale, la riduzione della latent size

consente di accorpare le feature di ogni item astraendo i concetti e riducendo per altro il carico computazionale.

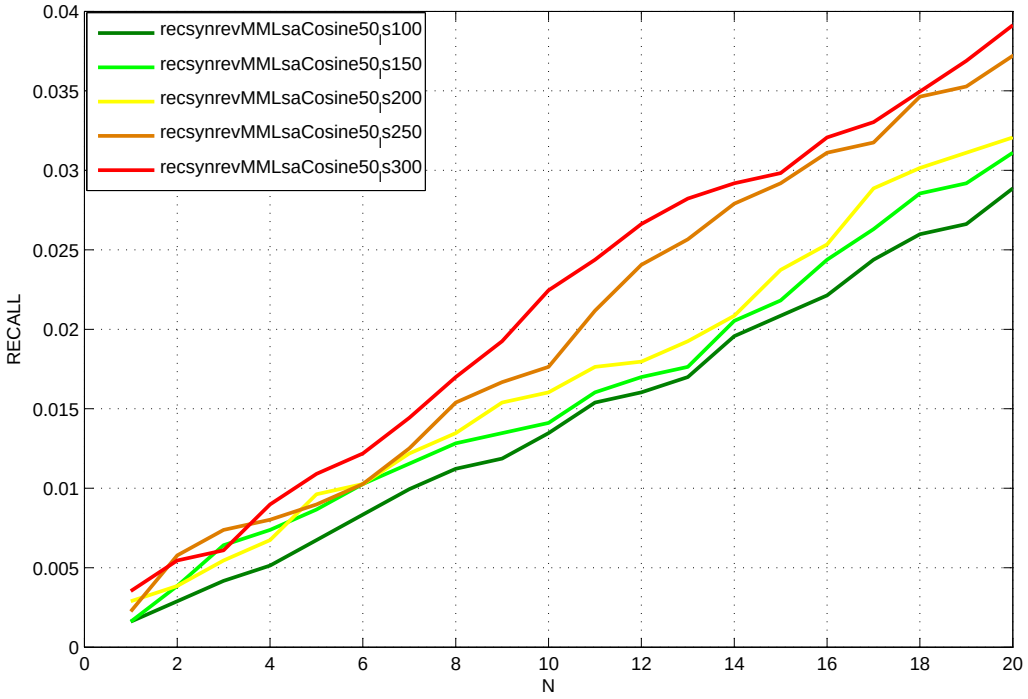


Figura 4.6: Curve della recall al variare della latent size su dataset MovieLens per l'algoritmo LsaCosine

Il parametro ls è quindi di fondamentale importanza per la qualità delle raccomandazioni prodotte dal sistema e i valori scelti, per questo algoritmo compresi tra 100 e 300 a step di 50, sono stati presi tra quelli più presenti in letteratura come per l'algoritmo Hydra. I risultati ottenuti su dataset MovieLens e mostrati in Figura 4.6 evidenziano come la soluzione migliore sia quella di utilizzare la latent size massima. Situazione analoga si ripete col dataset Yahoo! dove la soluzione scelta prevede di utilizzare ls pari a 250, come si evince dalla Figura 4.7.

Un altro parametro che governa il funzionamento dell'algoritmo è la variabile k del filtraggio *k-NN*, che conserva nel modello i k elementi più simili per ogni item. Anche per la scelta di quest'ultimo parametro sono stati utilizzati i valori più comuni pari a 50, 100 e 150. I risultati ottenuti dalla variazione non hanno però portato a particolari miglioramenti in alcuna configurazione, come è possibile osservare nell'esempio riportato in Figura 4.8 per il caso di raccomandazione sul dataset MovieLens. Si è quindi optato per una scelta del parametro pari al minimo, ovvero 50, poiché consente una in linea di principio una diminuzione del

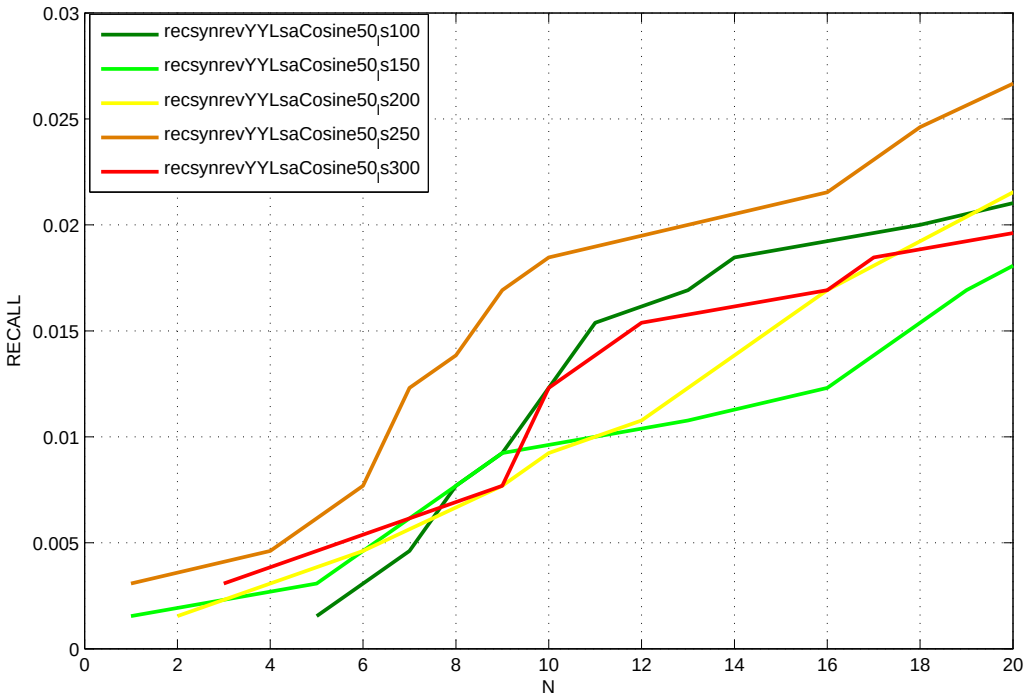


Figura 4.7: Curve della recall al variare della latent size su dataset Yahoo! per l'algoritmo LsaCosine

tempo di elaborazione così come di risorse impiegate.

Anche per questo metodo c'è stata una scelta indipendente per la versione della ICM da utilizzare dalla quale è emerso un significativo miglioramento nel caso di impiego della ICM estesa, come si evidenzia osservando i grafici in Figura 4.9. Da questo risultato emerge una peculiarità dei metodi basati su decomposizione SVD che è la predilezione per dataset molto grandi sui quali poter operare.

4.3.3 Metodi collaborativi

I metodi collaborativi hanno subito una profonda trasformazione da un punto di vista logico. Come discusso nel Paragrafo 3.5, l'impossibilità di creare un modello per la mancanza di informazioni è stata superata apprendendo il modello su un dataset differente da quello dove in seguito sono stati compiuti i test e valutate le performance.

L'ottimizzazione dei parametri che regolano i diversi algoritmi è stata compiuta sui due diversi dataset a differenza del caso content-based dove i dati utilizzati in fase di training sono condivisi. Assumendo, come per altro si verifica nella realtà, di non conoscere i parametri migliori da usare durante il testing sono stati utilizzati i valori ricavati per il training. Si è proceduto così con l'apprendere

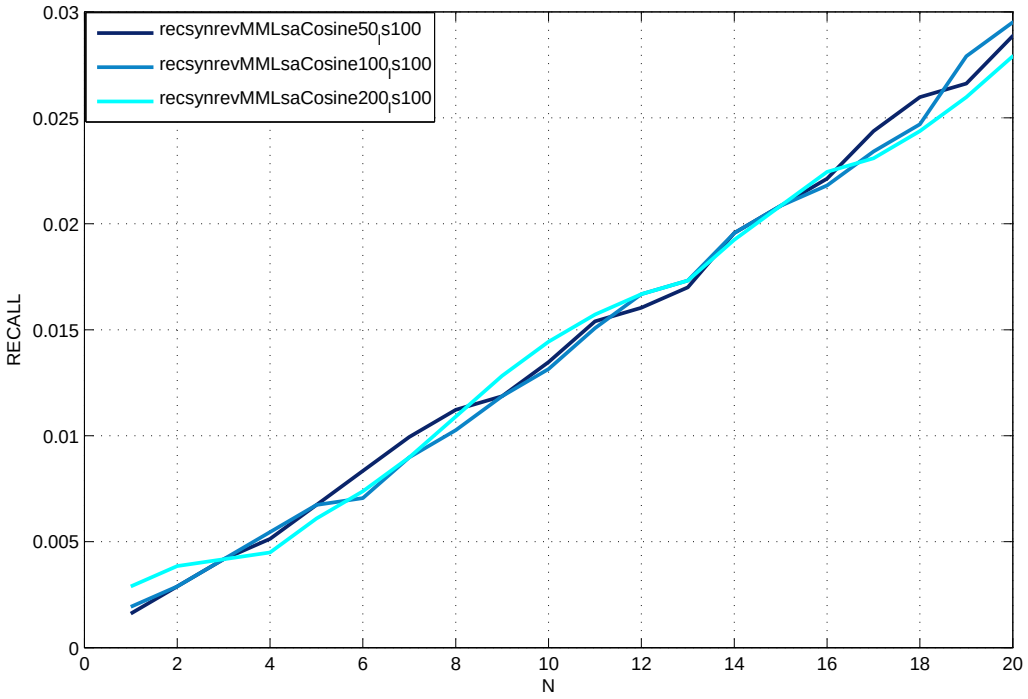


Figura 4.8: Curve della recall al variare del parametro k per l'algoritmo LsaCosine

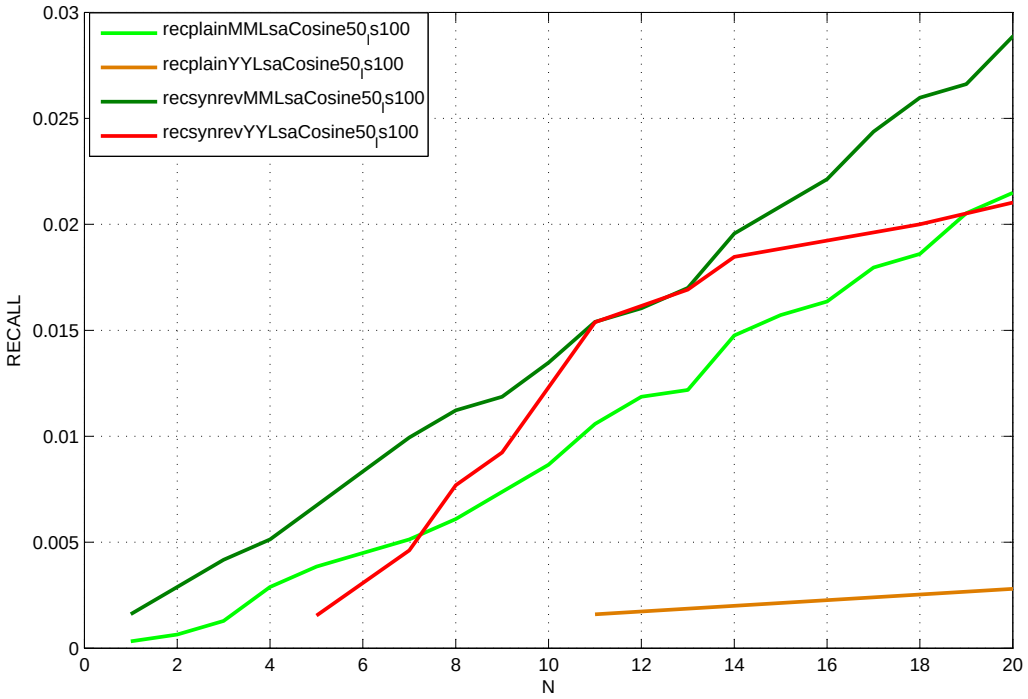


Figura 4.9: Curve della recall al variare della ICM per l'algoritmo LsaCosine

e testare un modello sul medesimo dataset al fine di costruire il caso ottimo di riferimento, per poi testare sullo stesso dataset un nuovo modello appreso da un dataset differente. Le possibili combinazioni realizzate sono riassunte in Tabella 4.3, dove le celle dalla matrice mostrano da quale dataset provengono i parametri in funzione del training set e del test set.

Questa scelta conservativa produrra risultati con una qualità inferiore rispetto al caso ottimo in cui siano noti per il dataset di destinazione. Proprio questo degrado è il fenomeno che si vuole osservare e valutare in questo lavoro di tesi per comprendere appieno se il porting di un algoritmo è possibile e a quale costo in termini qualitativi.

		test set	
		MovieLens	Yahoo!
training set	MovieLens	MovieLens	MovieLens
	Yahoo!	Yahoo!	Yahoo!

Tabella 4.3: Scelta dei parametri degli algoritmi in funzione dei dataset

Cosine k-NN

Il metodo Cosine valuta la similarità tra i diversi item, rappresentati in forma vettoriale, come l'angolo che essi individuano tra di loro, come descritto nel Paragrafo 2.6.3. L'aggiunta del filtraggio k-NN richiede una valutazione del miglior valore da utilizzare per ogni dataset al fine di massimizzare la qualità delle raccomandazioni.

Come mostrato in Figura 4.10, che raffigura a titolo di esempio i risultati sul dataset MovieLens, sono stati testati i valori più comuni di k per questo algoritmo pari a 50, 100 e 200. Sebbene la soluzione scelta con $k = 50$ fornisca una recall maggiore di almeno lo 0.5% per $N = 20$ rispetto alle altre curve, osservando il grafico è evidente come le alternative siano equivalenti. Effettuando il medesimo test sul dataset di Yahoo! si raggiunge lo stesso risultato. La decisione di optare per il valore minimo di k è stata suffragata dalla volontà di mantenere pochi item nella matrice di similarità così da ridurre il costo computazionale mantenendo invariata la bontà delle raccomandazioni.

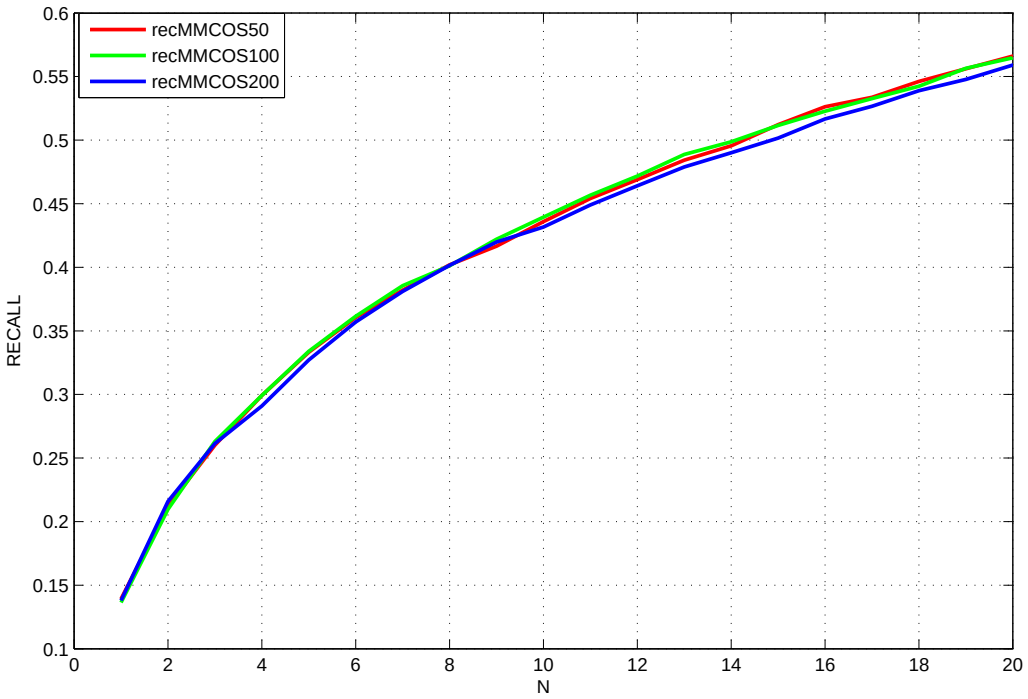


Figura 4.10: Curve della recall al variare del parametro k per l'algoritmo Cosine

Pearson k-NN

L'algoritmo Pearson valuta la differenza tra due diversi item utilizzando il coefficiente di correlazione di Pearson, come meglio descritto nel paragrafo 2.6.3. Così come per il metodo Cosine anche questo algoritmo fa uso del filtraggio k-NN al fine di rimuovere i contributi meno importanti. Come mostrato in Figura i valori usati in letteratura per il paramentro k in questo algoritmo sono 20 e 50.

Osservando il grafico che riporta le curve per entrambi i dataset, in rosso per MovieLens e in verde per Yahoo!, emerge come il miglior valore da assegnare alla variabile k sia 20 per il caso MovieLens mentre per quello Yahoo! sia pari a 50.

PureSVD

L'algoritmo PureSVD si basa sulla decomposizione SVD per la costruzione del modello, come descritto nel Paragrafo 2.6.3. La qualità della compressione dello spazio degli utenti al fine di creare delle categorie latenti dipende dalla scelta del numero di feature da conservare. Questo valore, noto come latent size o ls, è presente in letteratura per questo specifico algoritmo nelle tre alternative pari a 50, 100 e 150.

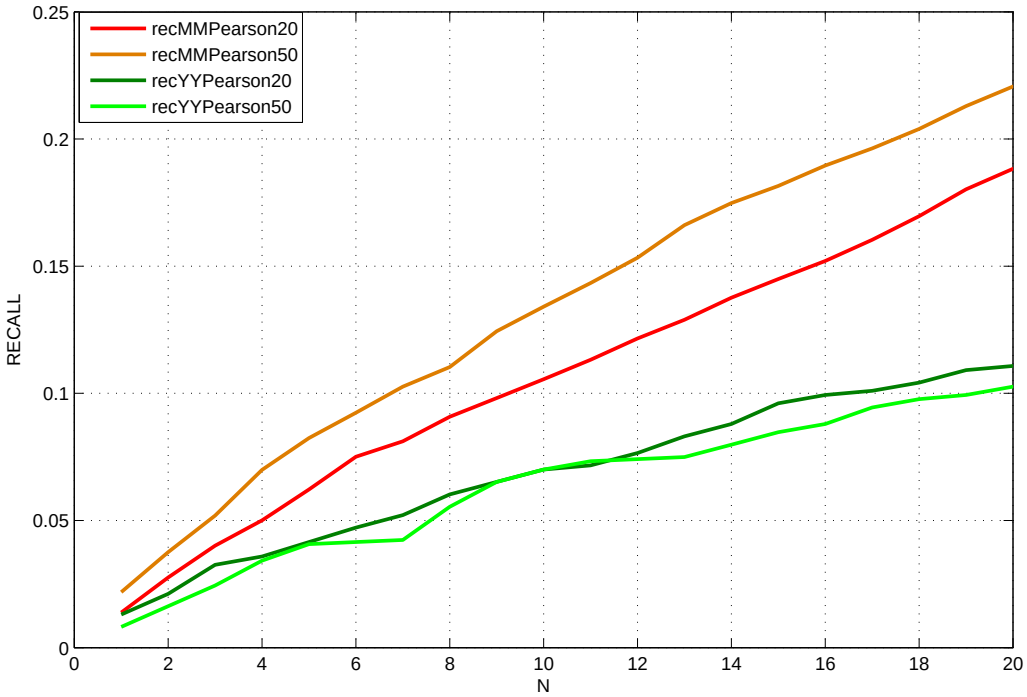


Figura 4.11: Curve della recall al variare del parametro k per l'algoritmo Pearson

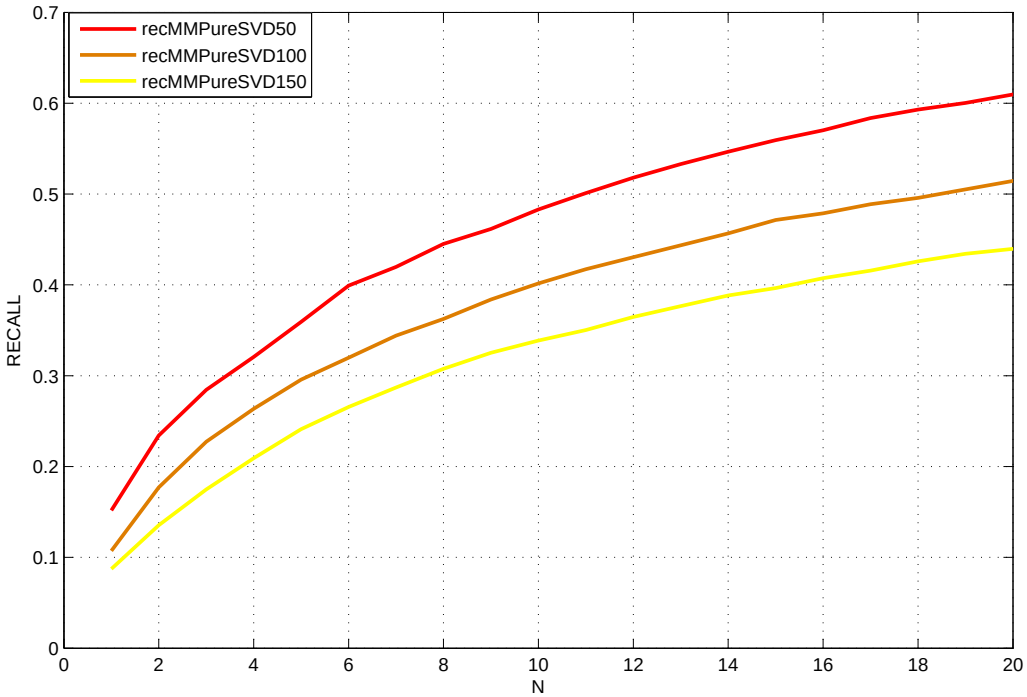


Figura 4.12: Curve della recall al variare del parametro ls per l'algoritmo PureSVD

Come è possibile notare dal grafico in Figura 4.12, che mostra il comportamento del metodo PureSVD al variare del parametro *ls* col dataset MovieLens, il miglior risultato si ottiene per il minimo valore della latent size, pari a 50. Un risultato analogo, con differenze più contenute, si evince dai test condotti sul dataset Yahoo!.

AsymmetricSVD

Gli strumenti alla base del metodo AsymmetricSVD sono il coefficiente di correlazione di Pearson e la decomposizione SVD, come meglio descritto nel Paragrafo 2.6. A differenza dei metodi ibridi che in generale separano in step differenti i diversi algoritmi, questo metodo propone un modello integrato che si avvantaggi delle peculiarità di entrambe le tecniche. Da un punto di vista prettamente algoritmico il metodo risulta molto complesso poiché sono stati introdotte un numero significativo di modifiche sostanziali così come di filtraggi portando il sistema ad essere regolato da numerosi parametri che incidono fortemente sulla bontà delle raccomandazioni.

La metodologia seguita per la scelta dei parametri è stata fedele al paper originale [34] così che sono stati imposti i valori di tutte le variabili fatte salve due per le quali è stata compiuta un'ottimizzazione. La prima esprime un learning rate mentre l'altra, λ, è un peso che regola la normalizzazione *baseline estimates*. Poiché l'algoritmo si basa sulla minimizzazione di una funzione obiettivo valutando l'errore quadratico medio (RMSE), per l'apprendimento dei parametri è stata utilizzata una discesa del gradiente per ognuno dei dataset alla ricerca dei valori che minimizzano l'errore.

In particolare partendo da un punto corrispondente alla soluzione ottima trovata nel paper, con *lrate* pari a 0.002 e λ pari a 0.04, si è proceduto variando il primo parametro in entrambe le direzioni in cerca di un minimo locale per poi passare ad ottimizzare con la stessa tecnica il secondo parametro. Infine il test è stato compiuto scambiando l'ordine delle variabili così da verificare tutti i casi possibili. Il numero massimo di passi consigliato dal paper per la discesa del gradiente è pari a 15 ma dai dati sperimentali sono emersi casi in cui sarebbero stati necessari ulteriori passaggi per cui il parametro è stato fissato a 20. Questo tipo di ottimizzazione è in grado di trovare un minimo locale senza altre garanzie poiché l'esplorazione dello spazio delle soluzioni è molto limitato. Non sono state implementate tecniche più sofisticate di discesa come *multiple restart* o *step decay* poiché l'obiettivo di questa ottimizzazione è stato quello di fornire valori dei parametri ammissibili che garantissero buoni risultati senza effettuare un tuning estremo.

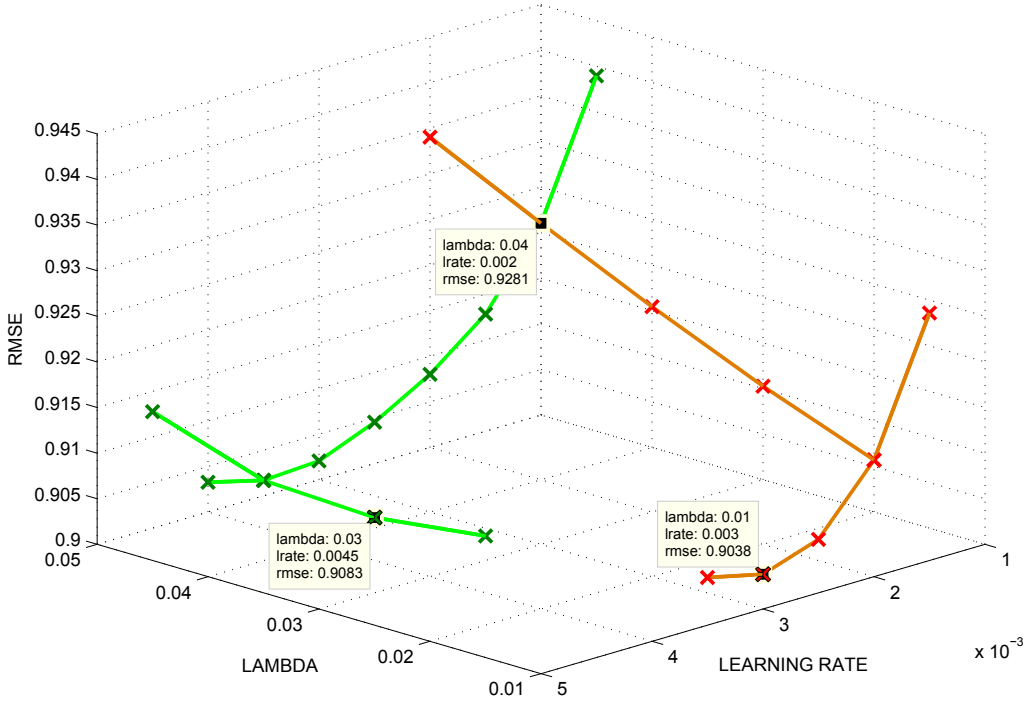


Figura 4.13: Discesa del gradiente per il dataset MovieLens per l'algoritmo AsymmetricSVD

Un esempio di discesa del gradiente sul dataset MovieLens è mostrato in Figura 4.13 dove la curva verde mostra come dapprima l'algoritmo ha tentato di aumentare il parametro *learning rate* trovando una soluzione peggiore per poi migliorare diminuendolo. Alla prima avvisaglia di peggioramento l'algoritmo torna nuovamente al passo precedente e a questo punto ricomincia l'ottimizzazione variando il paramentro λ, aumentandolo e poi riducendolo, per fermarsi nel punto con *lrate* pari a 0.0045 e λ pari a 0.03. Questa semplice ottimizzazione ha consentito di portare l'RMSE da 0.9281 a 0.9083.

In modo analogo si comporta la curva rossa che ottiene però un RMSE migliore e pari a 0.9038. Tutte gli altri casi non visualizzati che coprono ogni possibile altra combinazione, come riassunto in Tabella 4.4 dove in grassetto sono riportati i casi presenti nel grafico, si riconducono attraverso cammini diversi ai punti ottenuti nell'esempio.

Una medesima opera di ottimizzazione è stata compiuta sul dataset Yahoo! ottenendo i risultati riassunti in Tabella

step 1	parametro	λ	$\boldsymbol{\lambda}$	λ	λ	*lrate*	***lrate***	*lrate*	*lrate*
	verso iniziale	↑	↑	↓	↓	↑	↑	↓	↓
step 2	parametro	*lrate*	***lrate***	*lrate*	*lrate*	λ	$\boldsymbol{\lambda}$	λ	λ
	verso iniziale	↑	↓	↑	↓	↓	↓	↑	↓

Tabella 4.4: Casi testati nella discesa del gradiente su dataset MovieLens per l'algoritmo AsymmetricSVD

4.3.4 Metodi ibridi

Hydra

L'algoritmo Hydra, discusso nel Paragrafo 2.7, è stato scelto tra i diversi algoritmi ibridi poiché si fonda sulla decomposizione SVD che nelle altre classi di algoritmi garantisce una buona qualità dei risultati unita ad un'ottima performance in termini computazionali. Inoltre è l'unico metood ibrido a far uso delle informazioni presenti nelle tre matrici URM, ICM ed UCM. Per alleggerire ancor più il carico si è scelto di operare unicamente sulla versione ridotta della ICM ottenendo anche lo scopo di non favorire troppo i dati content a causa della grossa dimensione della ICM rispetto alle altre componenti. Indispensabile per la buona riuscita del metodo è la scelta accurata della latent size così come il peso assegnato ad ogni componente della Extended Rating Matrix. Si è così proceduto ad effettuare dei test su entrambi i dataset per valutare il contributo dei pesi così come la differenze introdotte dalla variazione della latent size.

Come valori della latent size sono stati considerati quelli più usati in letteratura pari a 50, 100 e 150 così da rendere i risultati comparabili. Si noti inoltre che in nessun paper si è fatto ricorso ad un apprendimento del parametro che viene fissato a priori in un insieme ridotto di valori [34, 19].

Per quanto concerne i pesi sono stati presi in esame tre diverse tipologie. La prima ha lo scopo di rendere uguali le medie dei valori delle tre matrici, come mostrato in Formula 4.1. Questa formulazione considera però solo gli elementi diversi da zero nel calcolo della media e questo potrebbe costituire un problema nel caso di matrici molto sparse.

$$\begin{aligned} \text{ICM=ICM} \times \frac{\text{mean(URM)}}{\text{mean(ICM)}} \\ \text{UCM=UCM} \times \frac{\text{mean(URM)}}{\text{mean(UCM)}} \end{aligned} \tag{4.1}$$

$$\begin{aligned} \text{mean}(X) = \sum_{i,j} \frac{X_{ij}}{\text{nnz}(X)} \\ \text{nnz}(Y) = |Y| \quad | \quad Y_{ij} \neq 0 \quad \forall i,j \end{aligned}$$

Per ovviare al problema sopraesposto si è pensato di utilizzare anche dei pesi che valutassero la sparsità delle matrici aumentato di fatto i pesi per le matrici più vuote così da enfatizzare l'effetto dei pochi elementi presenti. In Formula 4.2 sono riportate le funzioni che sono state utilizzate per questa normalizzazione.

$$\begin{aligned} \text{ICM=ICM} \times \frac{\text{sp(URM)}}{\text{sp(ICM)}} \\ \text{UCM=UCM} \times \frac{\text{sp(URM)}}{\text{sp(UCM)}} \end{aligned} \tag{4.2}$$

$$\text{sp}(X) = \frac{\text{nnz}(X)}{|X|}$$

Eguagliare le medie dei tre diversi insiemi consente comunque la presenza di valori massimi molto diversi tra le tre matrici per cui è stata implementata un'ulteriore alternativa in cui vengono resi uguali i valori massimi come mostrato in Formula 4.3.

$$\begin{aligned} \text{ICM=ICM} \times \frac{\text{max(URM)}}{\text{max(ICM)}} \\ \text{UCM=UCM} \times \frac{\text{max(URM)}}{\text{max(UCM)}} \end{aligned} \tag{4.3}$$

$$\text{max}(X) = X_{ij} \quad | \quad X_{ij} \geq X_{kl} \quad \forall k,l$$

Test su dataset MovieLens

Il primo test effettuato è stato quello di variare la latent size nel caso di pesi unitari. Dalle prove è emerso che il valore migliore per il parametro *ls* è 50, come

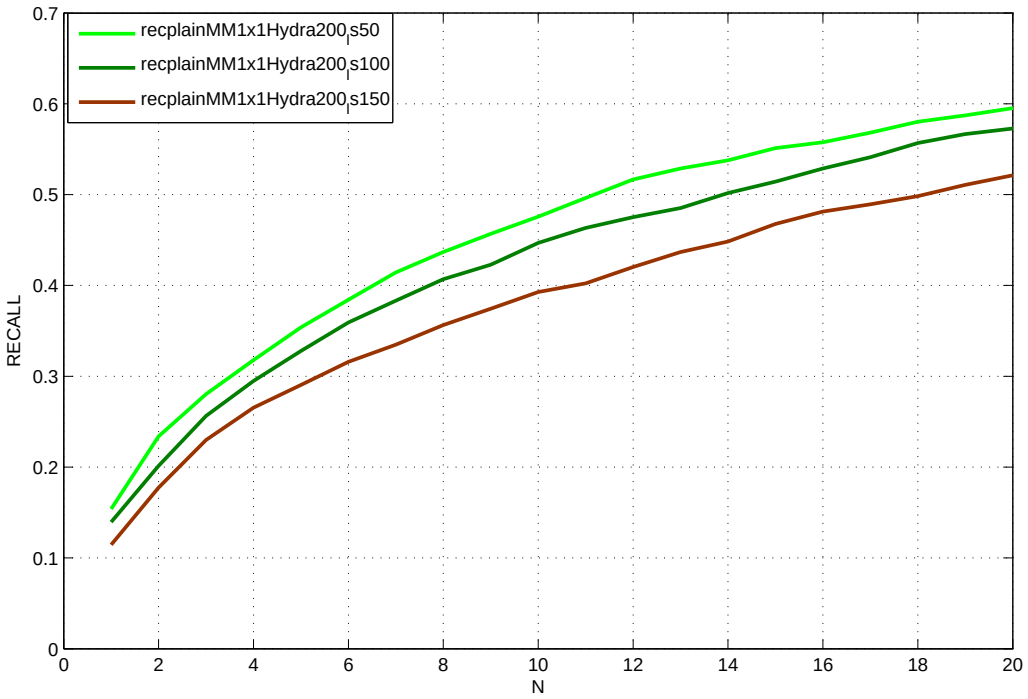


Figura 4.14: Curve della recall al variare della latent size, su dataset MovieLens per l'algoritmo Hydra

mostrato in Figura 4.14. A riprova del fatto sono stati compiuti ulteriori test con pesi casuali dal quale è emersa la medesima evidenza.

Nella prova successiva, una volta fissato il valore della latent size, sono stati testati i diversi pesi con cui normalizzare le matrici ICM ed UCM. In Figura 4.15 sono mostrati i risultati ottenuti dove la traccia rossa è il profilo originale senza l'uso dei pesi, quella verde è pesata con coefficienti in grado di rendere uguali i valori massimi delle matrici, la linea blu è stata ottenuta dall'uso di pesi in grado di rendere uguali le medie mentre quella gialla è stata ottenuta con coefficienti che tengono conto della sparsità di ICM e UCM rispetto a URM.

Alla luce dei test condotti non è possibile notare alcun miglioramento rispetto al caso base e, salvo il caso dei pesi legati alla media, nessun peggioramento.

Test su dataset Yahoo!

Anche sul dataset Yahoo! sono stati condotti i medesimi test. In Figura 4.16 sono riportati i grafici del caso base al variare della latent size tra i tre diversi valori scelti, dal quale emerge che il caso migliore è nuovamente quello che fissa il parametro ls pari a 50.

	URM	ICM	UCM
originale	1	1	1
massimi	1	9	109
medie	1	1028	10
sparsità	1	128	0.16

Tabella 4.5: Pesi calcolati per il dataset MovieLens

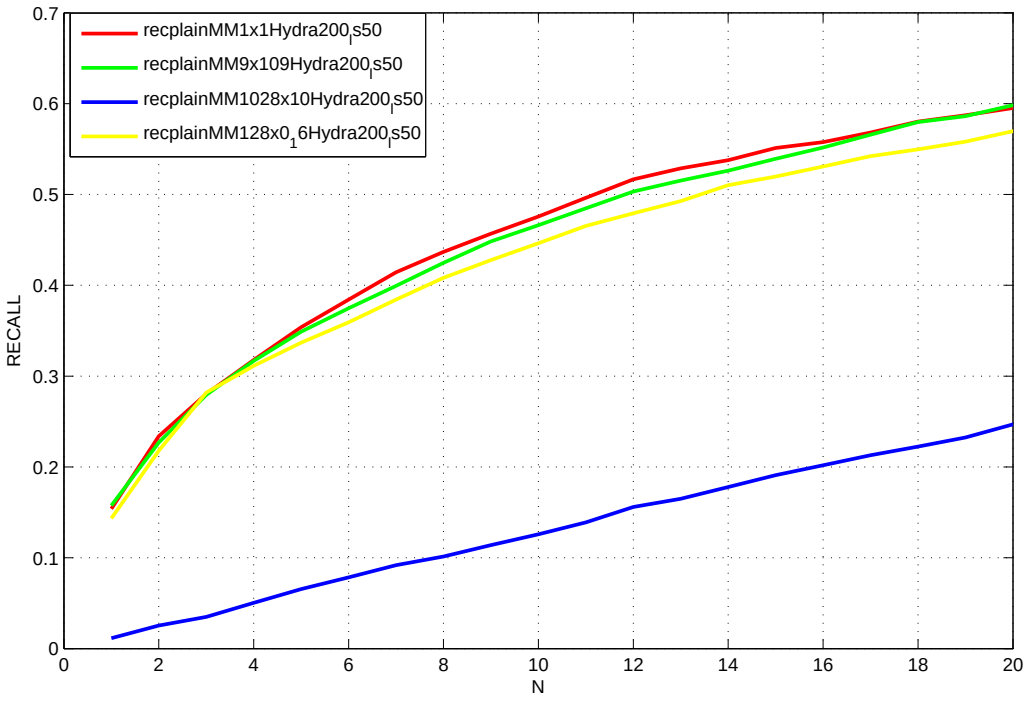


Figura 4.15: Curve della recall al variare dei pesi, su dataset MovieLens per l'algoritmo Hydra

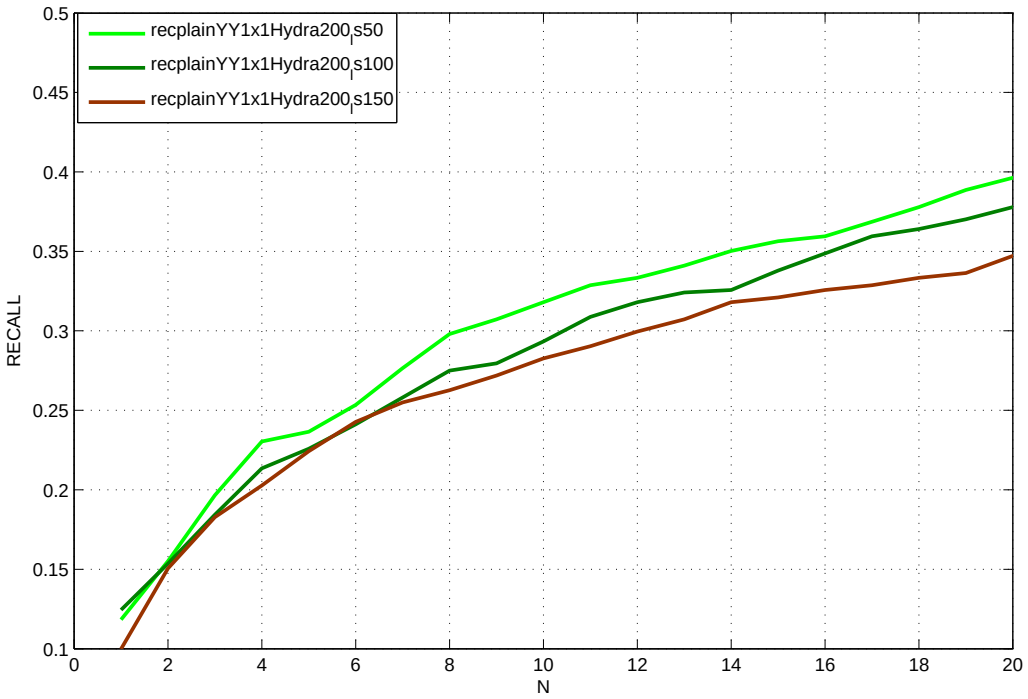


Figura 4.16: Le curve della recall al variare della latent size, su dataset Yahoo! per l'algoritmo Hydra

In Figura 4.17 sono mostrati i risultati ottenuti modificando i pesi per le diverse matrici utilizzando le stesse modalità del test su dataset MovieLens ma utilizzando i pesi riportati in Tabella 4.6. Anche in questo caso i risultati appaiono poco significativi fatta eccezione per la curva verde che mostra un miglioramento sul caso base del 3

	URM	**ICM**	**UCM**
originale	1	1	1
massimi	1	7	24
medie	1	50	0.6
sparsità	1	8	0.01

Tabella 4.6: Pesi calcolati per il dataset Yahoo!

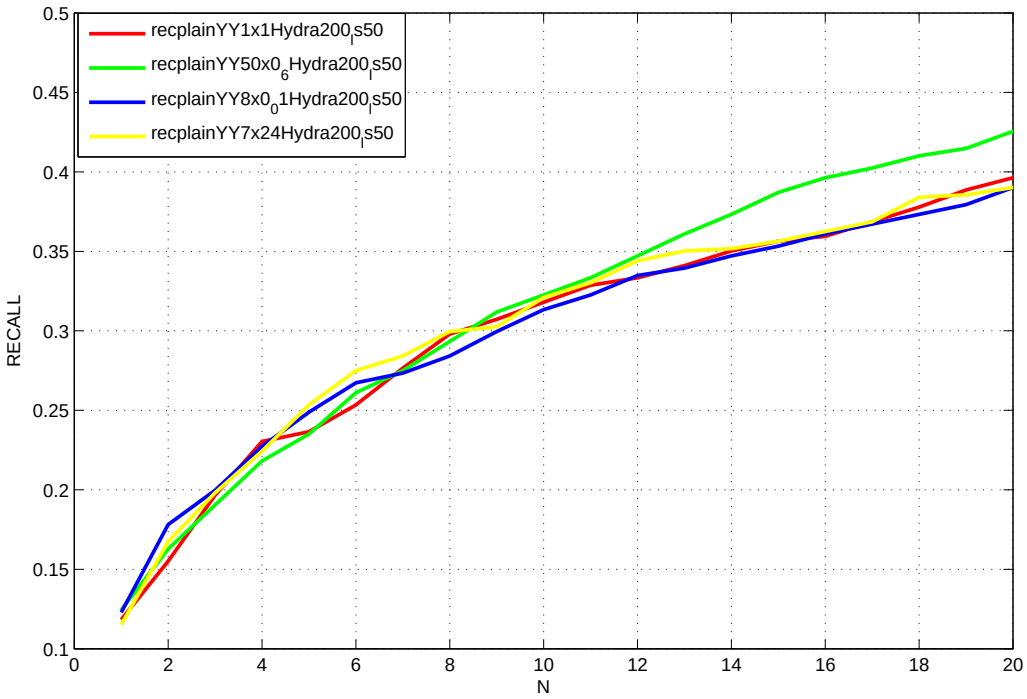


Figura 4.17: Curve della recall al variare dei pesi, su dataset Yahoo! per l'algoritmo Hydra

Considerazioni

Per meglio comprendere i risultati mostrati nei grafici è utile analizzare la Tabella 4.7 che riporta i valori delle tre diverse matrici dopo le normalizzazioni previste dall'algoritmo Hydra. I dati evidenziano una forte differenza tra tutte le grandezze che descrivono le matrici, fatto salvo per la riga *avg ≠0* che mostra l'effetto delle normalizzazioni. In particolare *densità*, *max* e *min* indicano rispettivamente la densità, il massimo valore e il minimo di ciascuna matrice. I valori etichettati come *avg all* indicano invece la media calcolata tra tutti gli elementi mentre *avg ≠0* indica la media tra tutti i valori diversi da zero. Infine *size vs URM* indica il rapporto tra il numero di elementi della matrice rispetto alla URM.

Alla luce di questi valori è possibile concludere che le matrici URM, ICM ed UCM di questi specifici dataset considerati siano troppo eterogenee per trovare dei pesi adeguati. Per questo il caso di miglioramento mostrato in Figura 4.17 può essere considerato un episodio di estremo overfitting che porterebbe poi a risultati fortemente compromessi in fase di porting. Considerando che questi risultati sono specifici per i dataset analizzati e per le metriche utilizzate per valutarli, si conclude che è preferibile lasciare i pesi pari a 1 come scelta conservativa.

	URM	**ICM**	**UCM**
densità	5.4%	0.0421%	33.33%
max	5.174	0.5773	0.0475
min	-3.5884	0.0031	0.0107
avg all	0.0504	0.00005	0.0048
avg $\neq 0$	1	1	1
size vs URM	1	6.5488	0.0022

Tabella 4.7: Analisi del dataset MovieLens dopo la normalizzazione compiuta dal metodo Hydra

Capitolo 5

Dataset

In questo capitolo vengono presentati i dataset utilizzati in questo lavoro di tesi per svolgere i test. Nel Paragrafo 5.1 vengono presentati i dataset MovieLens e Yahoo!, utilizzati per la seconda fase dei test su dataset reali. Durante la prima fase, invece, sono impiegati dataset artificiali costruiti a partire dal dataset reale Netflix, analizzato nel Paragrafo 5.2.

5.1 MovieLens e Yahoo!

MovieLens e Yahoo! sono dataset reali che contengono le valutazioni esplicite che gli utenti hanno espresso per i film così come le informazioni che li caratterizzano (titolo, regista, ecc.) ed hanno la caratteristica di condividere parte dei film. Sono inoltre presenti informazioni demografiche sugli utenti che hanno espresso i rating. Questi dataset sono pubblicamente disponibili e forniti rispettivamente dal laboratorio di ricerca GroupLens [28] e da Yahoo! Research [68].

5.1.1 Dati collaborativi

La parte che contiene le valutazioni, cioé la URM, viene fornita già divisa in due sottoinsiemi disgiunti, training set e test set, ma con caratteristiche differenti come mostrato in tabella reftab:dataset1. In particolare il rapporto tra il numero di rating del test set e del training set è fortemente sbilanciato tra i due dataset, essendo pari a 1.52% per MovieLens contro 4.8% per Yahoo!. Al fine di rendere comparabili i risultati ottenuti nei test dai due diversi dataset, i training set e test set per Yahoo! sono stati costruiti in modo da avere una proporzione pari a quella di MovieLens che è stato scelto come riferimento poiché già usato in diversi paper tra cui [19], [23] e [30].

Da questi dataset sono stati selezionati due sottoinsiemi, uno per ogni dataset, contenenti tutte e solo le valutazioni relative agli item condivisi, come esemplificato in Figura 5.1. La caratteristica più distintiva tra i due dataset, come mostrato in tabella 5.2, è la densità dei training set, che influenza fortemente la qualità del modello prodotto e quindi delle future raccomandazioni. Nel resto della tesi quando si farà riferimento ai dataset MovieLens e Yahoo! si intenderà sempre la porzione di dataset con item condivisi.

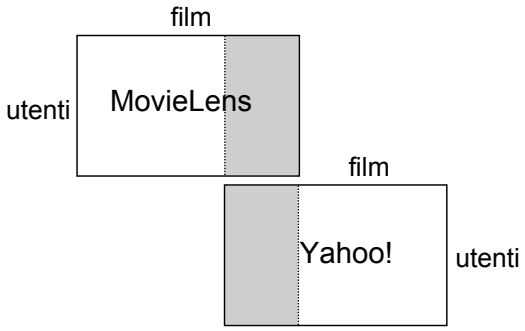


Figura 5.1: Selezione degli item comuni a MovieLens e Yahoo!

	MovieLens	**Yahoo!**
utenti	6040	7642
item	3883	11916
	training set	
rating	985205	218043
densità	4.2%	0.24%
	test set	
rating	15004	3321
test / training	1.52%	1.52%

Tabella 5.1: Struttura delle URM di MovieLens e Yahoo!

5.1.2 Dati content

La parte che contiene le caratteristiche dei film, cioé la ICM, è una matrice costruita a partire da un file che riporta le informazioni dei film organizzate in 33 categorie come titolo, sinossi, durata, ecc. Le informazioni ivi contenute sono state

	MovieLens	**Yahoo!**
utenti	6040	7642
item	2785	2785
	training set	
rating	909313	74482
densità	5.41%	0.35%
	test set	
rating	13856	1119
test / training	1.52%	1.50%

Tabella 5.2: Struttura dei sottoinsiemi estratti da MovieLens e Yahoo! con film condivisi

estratte, ripulite e raggruppate come spiegato nel Paragrafo 2.5 ed esemplificato in Figura 5.2. Si noti che la ICM è comune ai due dataset poiché assumiamo che sia semplice per un service provider ottenere questo genere di informazione.

In particolare per i test sono state costruite due ICM che, oltre a contenere i dati più importanti, differiscono per la presenza o meno della sinossi e del commento. Questa scelta è stata dettata dalla volontà di non sfavorire alcun algoritmo e di scegliere pertanto la miglior ICM per ogni singolo caso in esame. Le caratteristiche più salienti sono riportate in Tabella 5.3.

	ICM1	**ICM2**
metadati univoci	39555	66035
item	2785	2785
metadati totali	46335	218497
densità	0.04%	0.12%

Tabella 5.3: Struttura delle ICM dove ICM1 è la versione senza sinossi e commento mentre ICM2 è la versione completa

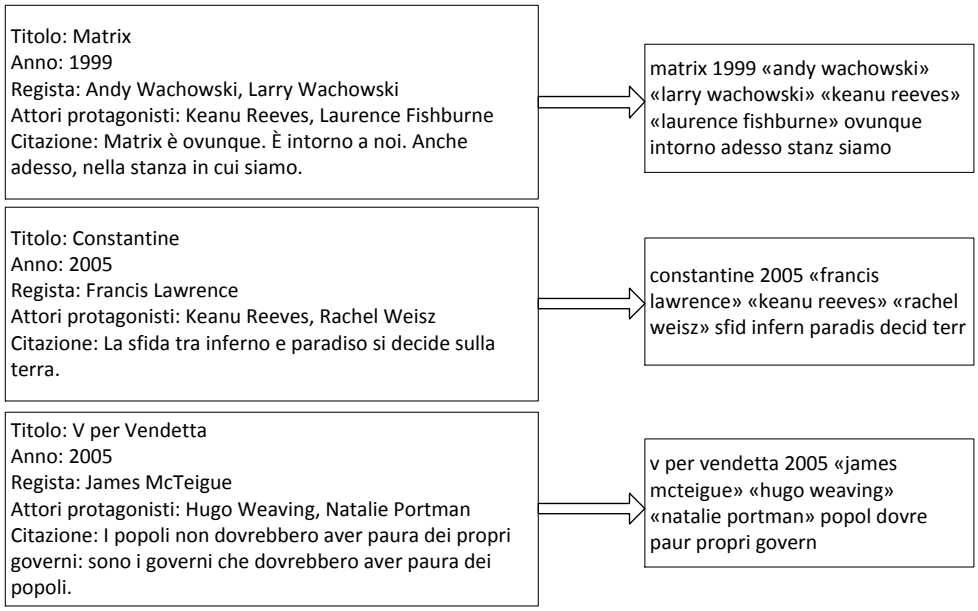


Figura 5.2: Esempio di costruzione della ICM

5.1.3 Dati demografici

Per questi due dataset sono disponibili anche informazioni aggiuntive relative agli utenti che vengono collezionate nella UCM. In particolare per il dataset MovieLens vengono forniti per ogni utente informazioni riguardo il genere, l'età, il tipo di occupazione e il codice postale identificativo della zona geografica. Poiché il dataset Yahoo! presenta solo informazioni relative a genere ed età, verranno considerati solo questi due attributi.

Il genere è espresso come un carattere dove M identifica il genere maschile ed F quello femminile. In totale nel dataset sono presenti 4331 uomini e 1709 donne, pari rispettivamente al 72% e al 28% del totale.

L'età viene fornità discretizzata in 7 intervalli come riassunto dalla Tabella 5.4 da cui emerge come la maggior parte degli utenti abbia un età compresa nella fascia tra 25 e 34 anni. Per avere un quadro generale più immediato i dati sono stati riportati nel grafico in Figura 5.3.

Per il dataset Yahoo! vengono fornite informazioni riguardanti il genere e l'anno di nascita dei diversi utenti.

Come per il dataset MovieLens anche in questo caso il genere è mappato utilizzando i caratteri m ed f. Analizzando le informazioni si evince che 5436 utenti sono uomini e 2183 sono donne, pari rispettivamente al 71.1% e al 28.5%. Il restante 0.4% degli utenti risulta non aver fornito l'informazione.

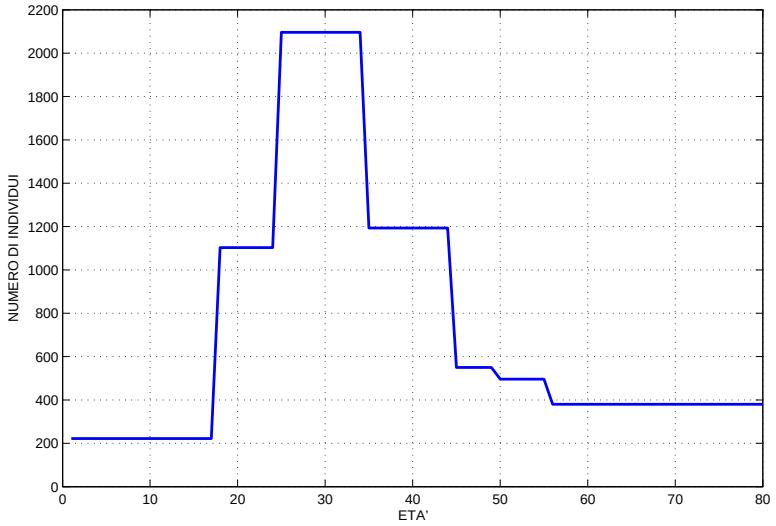


Figura 5.3: Suddivisione delle età in classi per il dataset MovieLens

classe	inizio	fine	individui	percentuale
1	1	17	222	3.7%
2	18	24	1103	18.3%
3	25	34	2096	34.7%
4	35	44	1193	19.7%
5	45	49	550	9.1%
6	50	55	496	8.2%
7	56	$+\infty$	380	6.3%

Tabella 5.4: Suddivisione delle età in classi per il dataset MovieLens

L'insieme degli anni di nascita è stato filtrato rimuovendo le date antecedenti al 1931, pari all'1% del totale, poiché molto probabilmente non corrispondono al vero. In Figura 5.4 viene riportato il grafico che mostra come la maggior parte degli utenti sia nata in un range molto ridotto di anni, compresi tra il 1980 e il 1987.

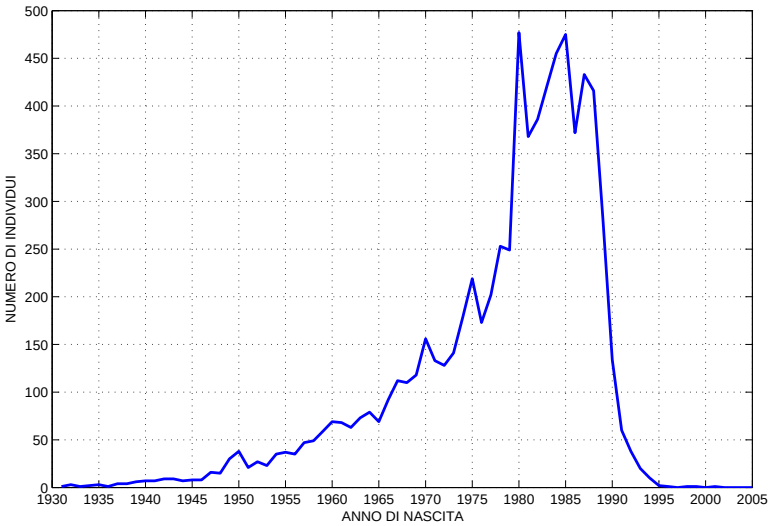


Figura 5.4: Suddivisione degli anni di nascita per il dataset Yahoo!

Al fine di rendere i dati provenienti dai due dataset comparabili tra loro sono stati mantenuti per entrambi le informazioni riguardanti l'età e il genere. Per quanto concerne le età queste sono state accorpate in quattro macro categorie definite in letteratura come *youngster*, *young*, *adult* e *old* [17]. Il mapping delle informazioni sulle nuove classi è mostrato in Tabella 5.5 da cui emerge come la classe più rappresentativa sia quella degli individui con età compresa tra 18 e 34 anni mentre quella che comprende i minorenni è la meno popolata.

classe	inizio	fine	percentuale MovieLens	percentuale Yahoo!
youngster	1	17	3.7%	0.2%
young	18	34	53%	67%
adult	35	55	37%	28.9%
old	56	$+\infty$	6.3%	3.9%

Tabella 5.5: Suddivisione delle età in classi per il dataset MovieLens

5.2 Netflix

Il dataset Netflix è un dataset reale esplicito, così come MovieLens e Yahoo!, ma contenente solo le valutazioni espresse dagli utenti ed è fornito da una delle più grandi compagnie statunitensi di noleggio e streaming di contenuti video.

Il dataset, le cui caratteristiche sono riportate in Tabella 5.6, viene fornito come gli altri già partizionato in training set e test set ma per la tipologia degli esperimenti condotti è stato mantenuto solo il training set, da cui sono stati estratti i nuovi training e test set. La motivazione che ha spinto a questa scelta è che il test set è stato probabilmente selezionato ad hoc per la competizione Netflix Prize [44] e questo potrebbe alterare in modo significativo i risultati.

	Netflix
utenti	480188
item	17770
rating	100462737
densità	1.18%

Tabella 5.6: Struttura delle URM di Netflix

Capitolo 6

Metodologia di test

In questo capitolo si affronta la metodologia utilizzata per lo svolgimento e la valutazione dei test. Nel Paragrafo 6.1 vengono presentate diverse famiglie di metriche per valutare la qualità delle raccomandazioni. Poiché per i metodi content-based non c'è accordo in letteratura su quali metriche siano da preferire, nel Paragrafo 6.2 viene compiuta un'indagine per stabilire quali impiegare in questo lavoro di tesi. Infine, nel Paragrafo 6.3, viene discusso come impostare i test al fine di ottenere risultati significativi evitando l'overfitting.

6.1 Metriche

Al fine di poter valutare la bontà delle raccomandazioni fornite dai diversi algoritmi è necessario stabilire una o più metriche in grado di esprimere le peculiarità dell'intera lista di raccomandazione in una grandezza numerica sintetica. In particolare è utile misurare come si discosta l'ordinamento degli item proposto dal sistema di raccomandazione con i valori reali che avrebbe assegnato l'utente [30].

Nell'ambito dei sistemi di raccomandazione esistono due famiglie principali di metriche basate una sull'errore e l'altra sulla classificazione.

6.1.1 Metriche di errore

Una metrica di errore in generale valuta la distanza tra il valore ottenuto da un esperimento e il valore atteso. Nello specifico caso dei sistemi di raccomandazione queste metriche misurano l'errore compiuto nella predizione per uno specifico utente dei rating associati ai diversi item con il valore reale noto. Più formalmente sia $\hat{r}_{ui}$ il rating stimato per l'utente u relativamente all'item i e sia r_{ui} il vero rating noto, la metrica di errore misurerà la distanza $err = dist(\hat{r}_{ui}, r_{ui})$ tra $\hat{r}_{ui}$ e r_{ui}.

Le metriche di errore più diffuse nella valutazione dei sistemi di raccomandazione sono RMSE (Root Mean Square Error), MSE (Mean Square Error) e MAE (Mean Absolute Error).

$$RMSE = \sqrt{MSE}$$
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{r}_{ui} - r_{ui})^2$$
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{r}_{ui} - r_{ui}|$$

Tutte queste metriche valutano la distanza tra il valore stimato e quello reale ma se MAE cresce linearmente al crescere dell'errore, MSE tende ad enfatizzare gli errori più grandi di un'unità crescendo come il quadrato della distanza mentre RMSE ha un profilo simile a MAE ma sempre maggiore.

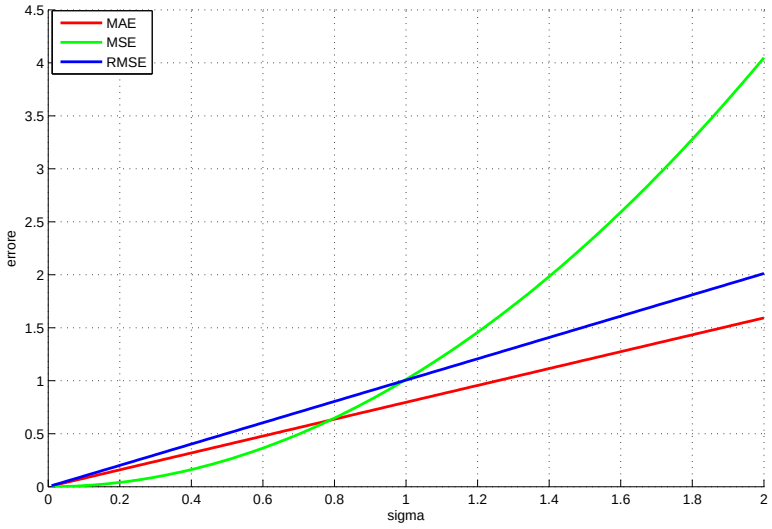


Figura 6.1: Esempio di valutazione dell'errore con le 3 metriche

In Figura 6.1 viene mostrato un esempio delle tre metriche a confronto testandole su una popolazione a cui è stato aggiunto un rumore gaussiano a media nulla di cui è indicata la deviazione standard via via crescente sull'asse delle ascisse.

Questa tipologia di metriche si è rivelata poco significativa per valutare algoritmi che si propongono di calcolare raccomandazioni top-N e risultano più indicate nel caso di raccomandazioni che tentino di stimare i rating in maniera precisa [30]. Inoltre la maggioranza dei paper che trattano l'argomento dei sistema di raccomandazione collaborativi fa uso di queste metriche per cui sarebbe difficile comparare i risultati ottenuti con quelli provenienti dalle metriche di errore. Per

questi motivi le metriche utilizzate in questo lavoro di tesi per valutare gli algoritmi collaborativi sono le metriche di classificazione, trattate di seguito. Per i metodi content-based è stata compiuta un'indagine più approfondita nel Paragrafo 6.2 poiché dalla letteratura non appare immediatamente evidente quale sia la scelta migliore.

6.1.2 Metriche di classificazione

Le metriche di classificazione misurano la capacità di un sistema di raccomandazione di separare correttamente gli item in fase di raccomandazione [55]. Da un punto di vista insiemistico un item può essere rilevante ed appartenere al sottoinsieme Ril dell'insieme di tutti gli item All oppure può essere non rilevante ed appartenere al sottoinsieme Irr. I due sottoinsiemi sono formalmente disgiunti e l'insieme di tutti gli item è dato dalla loro unione, come riassunto nella seguente formula.

$$Ril \cap Irr = \varnothing$$
$$Ril \cup Irr = All$$

Posta questa suddivisione, un sistema di raccomandazione selezionerà e proporrà all'utente un sottoinsieme Rac dell'insieme degli item All escludendo il sottoinsieme Esc degli item giudicati non interessanti per l'utente. La struttura dei due sottoinsiemi è analoga al caso precendente.

L'obiettivo della raccomandazione sarà quello di far coincidere il più possibile i sottoinsiemi Ril e Rac così come Irr ed Esc. Ogni eventuale discrepanza dal caso ottimo in cui si verifica una perfetta sovrapposizione degli insiemi costituirà quindi una misura della qualità del sistema di raccomandazione ed in particolare della sua capacità di svolgere il compito di classificare correttamente gli item. Si osservi che in caso di dataset espliciti oltre agli item rilevanti ed irrilevanti saranno presenti anche dei valori intermedi.

L'intersezione dei diversi sottoinsiemi, come mostrato in Figura 6.2, origina quattro diverse tipologie di item che dipendono sia dalla loro rilevanza che dal fatto che siano o meno stati inclusi nella lista di raccomandazione. Con *true positive* o TP vengono indicati tutti quegli item suggeriti all'utente e che questo trova rilevanti mentre con *true negative* o TN vengono identificati gli item esclusi dalla lista di raccomandazione e che non risultano interessanti per l'utente. Un sistema di raccomandazione reale introduce tuttavia una serie di errori di classificazione che possono essere ricondotti ai due sottoinsiemi *false positive* e *false negative*. Il primo, indicato in Figura 6.2 come FP, indica tutti quegli elementi che fanno

parte della raccomandazione ma che l'utente non gradisce mentre il secondo, noto come *FN*, raggruppa gli item esclusi dalla lista di raccomandazione ma che invece erano rilevanti per l'utente. Si noti che con le abbreviazioni si intende sempre indicare la cardinalità degli insiemi che rappresentano.

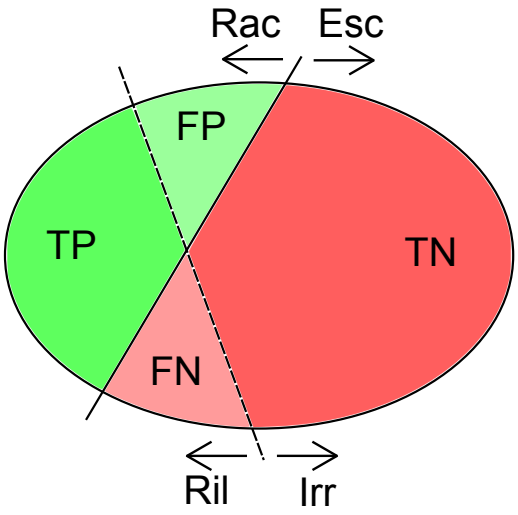


Figura 6.2: Suddivisione logica degli item nell'ambito delle metriche di classificazione

L'analisi dei rapporti tra le cardinalità di questi nuovi sottoinsiemi consente di evincere informazioni significative sulla qualità del sistema di raccomandazione sensibilmente diverse da quelle ottenibili dalle metriche di errore. In particolare le nuove grandezze introdotte sono:

1. *recall* o *sensitivity*, definita come il rapporto tra gli item raccomandati che sono rilevanti e tutti gli elementi rilevanti.

2. *fallout*, ottenuta come rapporto tra gli item raccomandati ma irrilevanti e tutti gli item irrilevanti.

3. *precision*, ricavata come rapporto tra gli elementi raccomandati che sono rilevanti e tutti quelli raccomandati

4. *specificity*, definita come rapporto tra gli item non raccomandati che sono irrilevanti e tutti gli item irrilevanti. É pari ad (1-fallout)

In modo più formale le definizioni sono le seguenti:

$$recall = \frac{TP}{TP + FN}$$
$$fallout = \frac{FP}{FP + TN}$$
$$precision = \frac{TP}{TP + FP}$$
$$specificity = \frac{TN}{FP + TN}$$

Si consideri ad esempio un insieme di 100 elementi di cui 30 di interesse per l'utente. Un sistema di raccomandazione produce da questo insieme una lista di suggerimenti composta da 20 elementi di cui solo 13 sono rilevanti per utente. Le dimensioni dei vari sottoinsiemi introdotti verrrebbero calcolate come:

$$Rac = 20$$
$$Esc = 80$$
$$TP = 13$$
$$FP = Rac - TP = 7$$
$$Ril = 30$$
$$Irr = 70$$
$$TN = Irr - FP = 63$$
$$FN = Ril - TP = 17$$

Da queste informazioni si otterrebbe che i valori per le metriche di classificazione sarebbero:

$$recall = \frac{TP}{TP + FN} = \frac{13}{13 + 17} = \frac{13}{30} = 43\%$$
$$fallout = \frac{FP}{FP + TN} = \frac{7}{7 + 63} = \frac{7}{70} = 10\%$$
$$precision = \frac{TP}{TP + FP} = \frac{13}{13 + 7} = \frac{13}{20} = 65\%$$
$$specificity = \frac{TN}{FP + TN} = \frac{63}{63 + 7} = \frac{63}{70} = 90\%$$

Osservando i dati dell'esempio si potrebbe concludere che il sistema di raccomandazione presenta un'ottimo fallout e, a dispetto di una modesta recall, garantisce una buona precision.

Poiché queste metriche vengono utilizzate per valutare algoritmi top-N è utile conoscere per quale specifico valore del parametro N siano state calcolate. A

tal fine spesso si indica con *nomeMetrica@N* il valore ottenuto da una specifica metrica per una raccomandazione topN, ad esempio *recall@20* indica il valore della recall per una raccomandazione top-20. Per valutare meglio il comportamento degli algoritmi si è soliti variare il valore del parametro N e valutare le singole metriche così da costruire una curva che mostri il legame tra la bontà degli algoritmi e la dimensione della finestra di raccomandazione.

Un'ulteriore misura della qualità di un sistema di raccomandazione consiste nella valutazione della *Receiver Operating Characteristic* (ROC). Questa grandezza mostra l'andamento della recall al variare del fallout a pari valori di N e viene rappresentata su un grafico dove sull'asse delle ascisse è riportato il fallout mentre sull'asse delle ordinate è riportata la recall. Il significato di questa metrica è valutare quale fallout bisogna tollerare per avere una recall desiderata e viceversa.

Inoltre relativamente alla curva ROC spesso viene utilizzata come indice di qualità l'area sottesa alla curva (AUC).

Questa tipologia di metriche ha dimostrato di essere più affidabile nella valutazione delle raccomandazioni collaborative rispetto agli altri metodi ed è pertanto stata scelta in questo lavoro di tesi [13].

6.1.3 Metriche rank-based

Esistono in letteratura anche metriche che utilizzano il posizionamento nella lista di raccomandazione, detto *rank*, come informazione per valutare la qualità del sistema. Se in generale queste tecniche vengono scarsamente considerate ci sono particolari situazioni in cui possono rispondere meglio a qualche specifica esigenza.

Una di queste metriche è la *Average Precision* ed è calcolata come una pesatura delle diverse precision, calcolate al variare di N in una raccomandazione top-N, con la rilevanza del rank ottenuto [43]. Formalmente è definita come:

$$AP = \frac{\sum_{r=1}^{N}(P(r) \times rel(r))}{TP + FN}$$

Dove con r è indicato il rank, N è il numero di item nella raccomandazione, *P(r)* è la precisione calcolata in una raccomandazione top-r e infine *rel(r)* è una funzione che esprime la rilevanza del rank.

In caso di prove ripetute viene utilizzata anche la *Mean Average Precision* che è definita come la media delle Average Precision sul numero di prove effettuate. La definizione formale è la seguente.

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

Dove il parametro Q è il numero di prove.

Un'altra metrica che fa uso del rank è la *Discounted Cumulative Gain* (DCG) che si pone l'obiettivo di pesare l'utilità degli item in funzione del loro rank [16].

$$DCG_p = rel_1 + \sum_{i=1}^{p} \frac{rel_i}{\log_2 i}$$

Dove il parametro rel_i esprime la rilevanza dell'item al posto i mentre p è il massimo rank considerato.

Infine la metrica *Normalized Discounted Cumulative Gain* è una versione normalizzata della DCG ottenuta come rapporto tra la DCG e la DCG ideale. Lo scopo di questa normalizzazione è contenere i valori tra 0 e 1.

6.2 Valutazione dei metodi content-based

Se per la valutazione dei metodi collaborativi la scelta delle metriche da impiegare appare più limitata e semplice [13], per i metodi content-based non emerge chiaramente una metrica da utilizzare per valutare gli algoritmi e pertanto è stata compiuta una vasta indagine al fine di uniformare la scelta ai principali paper sull'argomento. I risultati più notevoli ottenuti nei diversi studi vengono di seguito riportati insieme alle metodologie impiegate.

Enhanced vector space models for content-based recommender systems. [43]

Il dataset impiegato in questo paper è MovieLens, che presenta valutazioni comprese tra 1 e 5 mentre l'assenza di rating è indicata con lo zero. Ai fini della raccomandazione i rating da 0 a 2 sono considerati negativi mentre quelli da 3 a 5 positivi. L'algoritmo impiegato è della famiglia content-based e risulta essere nuovo in letteratura (Random Indexing). I test sono stati effettuati con cross validation a 5-fold. Infine la metrica utilizzata è Average Precision@N con N ad indicare che le raccomandazioni sono del tipo top-N. I valori di N impiegati per questo studio sono i seguenti valori: 1,3,5,7,10. In Tabella 6.1 vengono riportati i risultati più rilevanti ottenuti dall'algoritmo Random Indexing per ciascun valore di N.

Content-based recommendation in social tagging systems. [16]

I dataset impiegati per questo studio sono quelli di Delicious [67], che contiene segnalibri di pagine web, e Last.fm [37], che si occupa di musica. Per costruire il test set è stato impiegato il 20% dei dati disponibili. Non è dato conoscere il range di valori assunti dai rating anche se è intuibile che siano compresi tra 1 e 5, poiché tutti i rating superiori a 2 sono stati considerati positivi mentre gli

Metrica	Result
AV-P @1	0.8593
AV-P @3	0.8578
AV-P @5	0.8575
AV-P @7	0.8561
AV-P @10	0.8545

Tabella 6.1: Average precision dell'approccio Random Indexing

altri negativi. L'algoritmo testato in questo paper è un coseno content-based con filtraggio tf-idf e impiego di cross validation con 5 fold. Le metriche impiegate per valutare i risultati sono Precision@N, MAP e DCG. Vengono infine riportati in Tabella 6.2 i principali risultati ottenuti per ciascuna metrica selezionata.

Metrica	Result
P @5	0.292
P @10	0.212
P @20	0.144
MAP	0.115
nDCG	0.350

Tabella 6.2: Risultati ottenuti sui dataset Delicious e Last.fm

Open user profiles for adaptive news systems: help or harm? [2]

Questo paper fa uso di un dataset di notizie di cui però non è fornita la provenienza. Le valutazioni appartengono a un range da 1 a 3 dove 0 rappresenta la mancanza di informazione. Gli item con rating pari a 3 sono considerati rilevanti mentre quelli irrilevanti hanno rating uguale a 0. Non vengono rilasciate ulteriori informazioni sugli algoritmi testati, a parte che si tratta di soluzioni contet-based. La metrica invece impiegata in questo lavoro è Average Precision@N. I valori ottenuti al variare di N sono riportati in Tabella 6.3.

Metrica	Result
AV-P @10	0.98
AV-P @20	0.91

Tabella 6.3: Average precision per il dataset di notizie

Intimate: a web-based movie recommender using text categorization. [40]

Questo studio impiega i dataset EachMovie [28] e IMDb [32], entrambi relativi al contesto cinematografico. L'algoritmo testato è dato dall'unione dell'approccio content-based con filtraggio k-NN con un classificatore bayesiano naïve. Nei test è impiegata la cross validation con 10 fold e le metriche recall e precision. Di seguito, in Tabella 6.4, vengono riportati i valori ottenuti per ciascuna metrica.

Metrica	Result
Accuracy	0.59
Recall	0.63
Precision	0.62

Tabella 6.4: Risultati ottenuti sui dataset EachMovie e IMDb

Content-boosted collaborative filtering for improved recommendations. [42]

Il dataset impiegato per questo studio è EachMovie, che presenta valutazioni comprese tra 1 e 5. I rating 4 e 5 sono considerati positivi mentre gli altri negativi, compreso lo zero che indica la mancanza di informazioni. Gli algoritmi impiegati in questo paper sono un metodo content-based Naïve Bayes e un collaborativo k-NN Pearson. Le prove sono state compiute con cross validation a 10 fold e le metriche impiegate sono MAE e ROC-4, dove il numero indica la soglia per considerare un rating positivo. In Tabella 6.5 sono riportati i valori conseguiti dai due diversi metodi per ciascuna metrica.

Algoritmo	MAE	ROC-4 area
content-based	1.059	0.6376
collaborative	1.002	0.6423

Tabella 6.5: Risultati ottenuti sul dataset EachMovie

CinemaScreen recommender agent: combining collaborative and content-based filtering. [54]

Il paper fa uso del dataset MovieLens sperimentando una soluzione custom di raccomandazione content-based ed una collaborativa basata su Pearson k-NN. Non vengono forniti ulteriori dettagli sulle prove effettuate se non per le metriche utilizzate: precision e recall.

Content-based Dimensionality Reduction for Recommender Systems. [63]

Questo lavoro impiega i dataset MovieLens e IMDb, considerando i rating con valore superiore a 3 positivi mentre gli altri negativi. Gli algoritmi valutati sono un content-based basato sul coseno e un collaborativo non specificato con filtraggio k-NN. Infine le metriche impiegate sono precision e recall. In Tabella 6.6 sono indicati i valori raggiunti da ciascun metodo, dove per ogni valore di recall è riportata la corrispondente precision.

	Recall	Precision
collaborative	0.01	0.55
	0.13	0.39
content	0.03	0.2
	0.05	0.11

Tabella 6.6: Risultati ottenuti sui dataset EachMovie e IMDb

6.2.1 Considerazioni

Dall'analisi dei dati si evincono due fatti principali. I metodi content-based, poiché vengono implementati e testati in condizioni differenti tra di loro, ottengono risultati fortemente contrastanti. Si considerino ad esempio i valori di precision in 6.3 e in 6.4. Gli esiti così distanti mostrano come diverse interpretazioni delle metriche, per altro non sempre pubblicate e quindi riproducibili, portino a risultati non comparabili.

Inoltre l'uso di metriche o stime dell'errore convenzionali nella valutazione di metodi content e collaborative portano comunque a preferire questi ultimi perché ottengono performance migliori anche se non c'è una chiara evidenza della loro superiorità nei casi reali. Per un esempio si osservino i valori di MEA e ROC in 6.5.

Sulla base dei documenti analizzati sono state scelte come metriche per i test condotti in questo lavoro Recall@20, Fallout@20 e AUC. Quest'ultima rappresenta l'area sottesa alla curva ottenuta ponendo il fallout sull'asse delle ascisse e la recall sulle ordinate.

6.3 Validazione dei risultati

L'apprendimento di un modello a partire dai dati è solo una parte del lavoro necessario per costruire un buon sistema di raccomandazione poiché lo scopo ultimo non è quello di replicare i dati osservati in passato ma compiere previsioni su dati mai visti primi. Si prenda l'esempio seguente, puramente didattico, che lega una variabile di ingresso ad una di uscita attraverso una funzione incognita da modellizzare.

$$f(0) = 0$$
$$f(1) = 1$$

A fronte di due sole osservazioni possono essere realizzate molteplici ipotesi per spiegare il fenomeno, ad esempio $f(x) = x$ oppure $f(x) = true(x)$ dove $true(x) = 0$ se $x = 0$ altrimenti $true(x) = 1$. Note un numero maggiore di osservazioni che non avvalorino i modelli più semplici nascerà la tentazione di costruire una funzione più complessa per trovare accordo con i dati sperimentali. A questo punto è lecito domandarsi se la funzione ideata sarà quindi in grado di modellizzare il fenomeno e quindi di avere capacità previsionali oppure sarà solamente capace di replicare i dati osservati. Questo problema noto come *overfitting* si presenta ovunque sia necessario un apprendimento automatico e deve essere identificato e ridotto per dare significatività al modello ottenuto.

Una delle strategie utilizzate per ridurre l'effetto dell'overfitting è nota come *cross-validation* e si basa sull'idea di apprendere il modello su una partizione dei dati per poi effettuare una fase di testing sulla restante parte. In un contesto come quello dei sistemi di raccomandazione la divisione in partizioni è compiuta in modo tale da dividere gli utenti in due sottoinsiemi disgiunti, come mostrato in Figura 6.3. La scelta di dividere il dataset in due parti e di non generare nuovi valori è spesso dettata da problemi reali come nei sistemi di raccomandazione dove non è possibile produrre nuovi dati poiché le osservazioni sono spesso forniti da aziende o gruppi di ricerca. Anche questa strategia è tuttavia prona ad errori poiché un singolo test non può essere considerato un indice affidabile riguardo la qualità del modello.

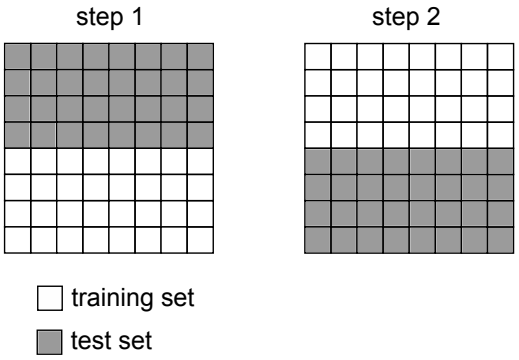


Figura 6.3: Esempio di K-fold cross validation con K=2

Per ovviare a questo problema è stata ideata una modalità chiamata *K-fold* cross-validation che prevede la divisione del dataset originale in K partizioni, dette fold, così che il modello possa essere appreso su $K-1$ partizioni mentre l'ultima viene utilizzata per il testing. Questo procedimento viene quindi iterato K volte in modo che tutti i fold vengano utilizzati una volta in fase di test. I risultati ottenuti vengono quindi mediati per considerare il singolo contributo di ciascun esperimento. É importante che in tutte le fasi di selezione e partizionamento dei dati le scelte siano fatte in maniera totalmente casuale così da fugare ogni possibile rischio di correlazione tra i dati come ad esempio serie cronologiche.

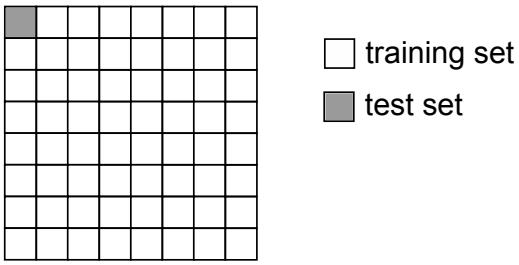


Figura 6.4: Esempio di leave-one-out

Nel caso in cui la variabile K venga posta uguale al numero di osservazioni, tale per cui i fold contengano un solo dato, la tecnica viene chiamata *leave-one-out* cross-validation. Un esempio di questa strategia è mostrato in Figura 6.4. Questa tecnica ha la forte limitazione dell'elevato costo computazione poiché dovrà creare tanti modelli quanti sono i dati del dataset.

Infine un'ulteriore tecnica utilizzata e nota come *holdout* consiste nel selezionare una parte dei rating selezionati a caso, e quindi non appartenenti a un sottoinsieme degli utenti come nel caso K-fold, e apprendere su questi il modello. Il testing verrà infine compiuto sulla parte restante dei rating, come rappresentato in Figura 6.5. Questa tecnica però non risolve efficacemente il problema dell'overfitting

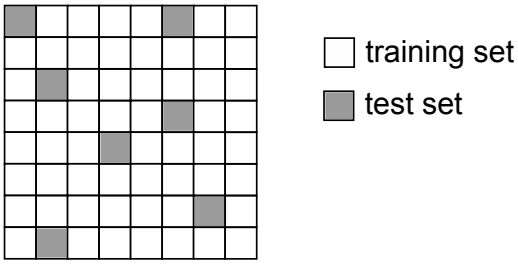


Figura 6.5: Esempio di holdout

e può inoltre alterare i risultati poiché il modello viene appreso su un dataset modificato e non solo ridotto.

In questo lavoro di tesi si è optato per una scelta ibrida che unisce la 2-fold cross-validation con la leave-one-out poiché questa metodologia si sta affermando come strategia dominante nella validazione degli studi sui sistemi di raccomandazione [19, 13, 20, 34].

Enhanced leave-one-out

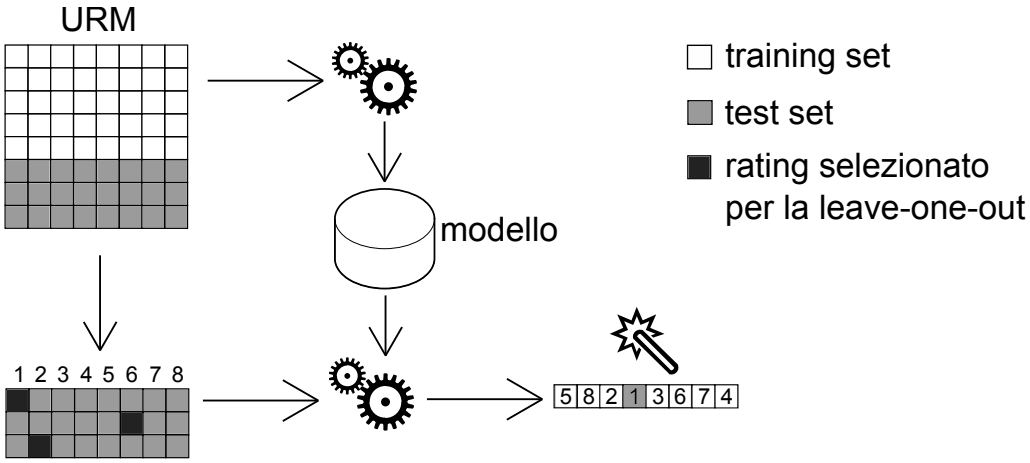


Figura 6.6: Schema di funzionamento della Hybrid leave-one-out

Questa tecnica è nata per unire le peculiarità della k-fold cross-validation e della leave-one-out. In particolare l'idea di separare training e test set è presente e mutuata dalla k-fold mentre la scelta casuale di item da rimuovere è stata mantenuta nella fase di testing ispirandosi alla leave-one-out originale. Uno schema esemplificativo è mostrato in Figura 6.6.

L'algoritmo comincia col dividere l'intero dataset in due partizioni disgiunte pari al 1.5% del totale per il test set mentre la restante parte costituisce il training set, sul quale verrà appreso il modello. Quando si è interessati al calcolo della recall,

e quindi è importante capire come vengono raccomandati gli item che l'utente gradisce, vengono selezionati dal test set una porzione di rating col valore massimo per una quantità pari a 2000 unità. A questo punto per ogni singolo rating estratto viene compiuta una raccomandazione per osservare in che posizione della lista viene posizionato l'item a cui il rating si riferisce, rimuovendo realisticamente gli item già utilizzati dall'utente. É plausibile che se il sistema di raccomandazione funzioni correttamente e quindi il modello appreso sia in grado di compiere delle previsioni affidabili l'item analizzato sarà nelle prime posizioni della lista. Nel caso invece in cui si fosse interessati al calcolo del fallout bisognerebbe selezionare dal test set un insieme di rating con valore minimo per poi verificare nella lista di raccomandazione che si trovino realmente agli ultimi posti.

Questa tecnica oltre a limitare fortemente il fenomeno dell'overfitting è anche conservativa dal punto di vista dei risultati forniti. Questo perché, ad esempio per un caso di recall, un item di interesse per l'utente e correttamente riconosciuto dal sistema di raccomandazione potrebbe però essere anticipato nella lista di raccomandazione da altri item anch'essi interessanti. Se il filtraggio top-N sulla lista così costruita fosse troppo selettivo si correrebbe il rischio concreto di escludere l'item oggetto di test dalla raccomandazione finale generando un riscontro negativo anche se gli item suggeriti fossero tutti rilevanti per l'utente. Anche il problema relativo al massiccio costo computazionale del metodo leave-one-out originale risulta mitigato perché il modello viene generato una sola volta per tutti i test che vengono compiuti.

Capitolo 7

Analisi su dataset artificiali

In questo capitolo sono riportati i risultati più importanti ottenuti dalla fase preliminare di test (vedi Paragrafo 4.1), nella quale sono stati impiegati i metodi collaborativi, descritti nel Paragrafo 2.6, su dataset artificiali costruiti ad hoc a partire dal dataset reale Netflix. Il dataset originale Netflix è stato suddiviso in due parti per simulare l'esistenza di due domini (target ed ausiliario). Sono state in seguito applicate le modalità di partizionamento descritte nel Paragrafo 4.1 valutando gli algoritmi al variare del numero di utenti e della densità dei rating del dominio ausiliario. Gli algoritmi utilizzati in questa fase sono:

- MovieAVG: Movie Average, non personalizzato.
- TopRated: Top Popular, non personalizzato.
- COS100: item-based collaborativo, Cosine con parametro k-NN = 100.
- Pearson20: item-based collaborativo, Pearson con parametro k-NN = 20.
- PureSVD50: item-based collaborativo, PureSVD con 50 fattori latenti.
- AsySVD100: item-based collaborativo, Asymmetric SVD con 100 fattori latenti.

I risultati di qualità calcolati sul dominio target sono presentati, rispettivamente, nei Paragrafi 7.1 e 7.2.

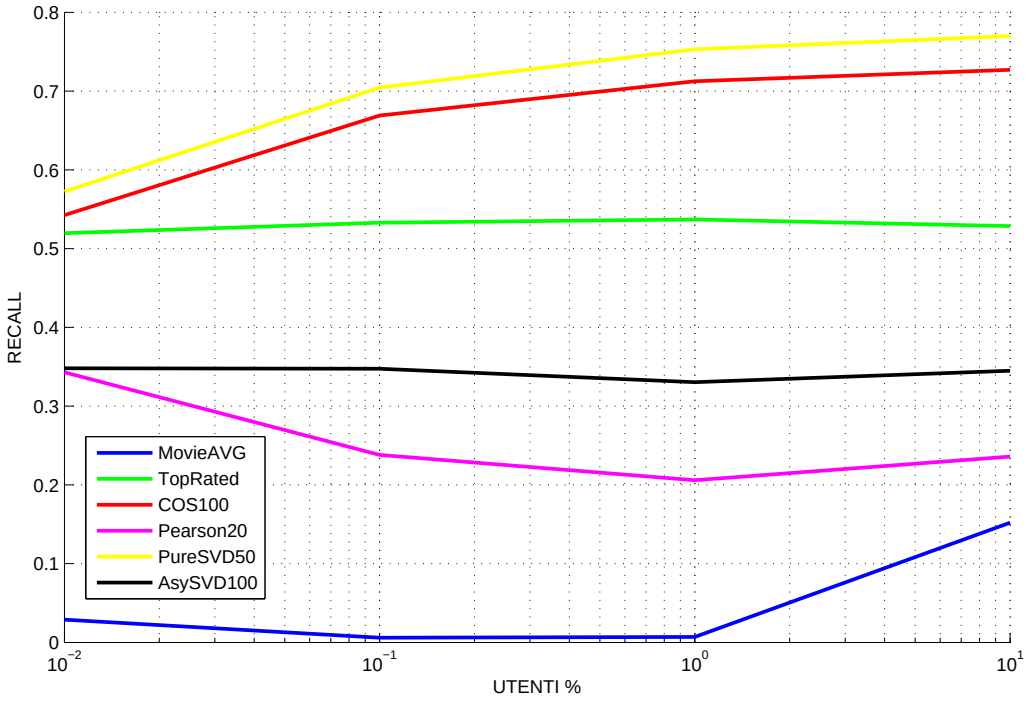
7.1 Analisi sulla variazione del numero di utenti

In questo paragrafo vengono presentati e discussi i risultati ottenuti dai test preliminari compiuti per valutare l'influenza del numero di utenti presenti nel training set sulla bontà delle raccomandazioni.

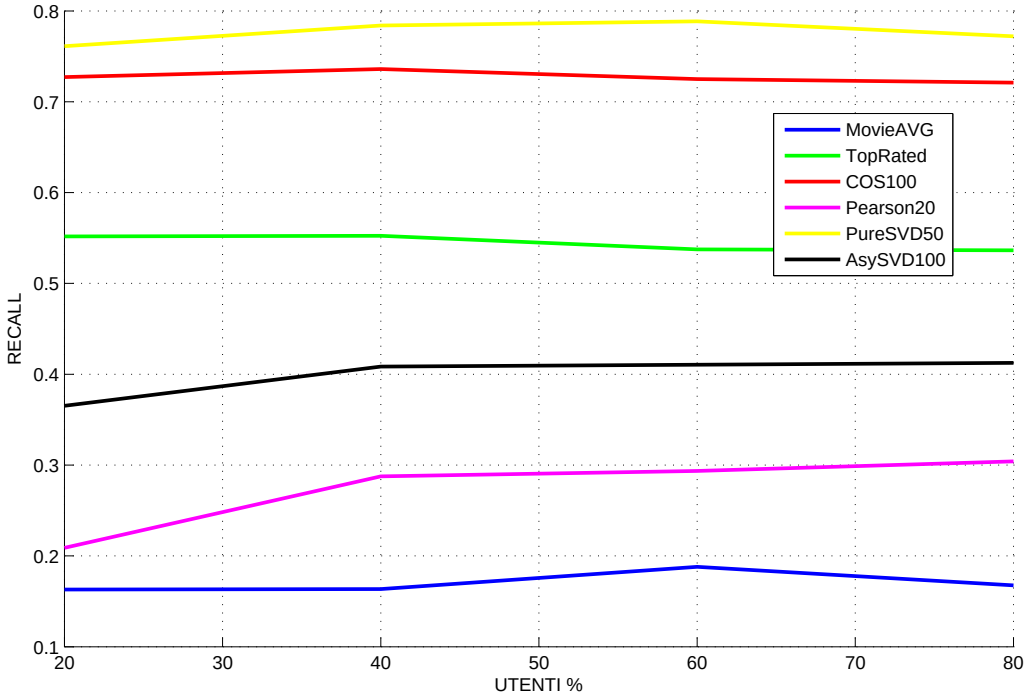
Per questa tipologia di risultati vengono presentati due grafici, uno per i valori più piccoli del numero degli utenti mentre l'altro per i più grandi. Ciascun grafico mostra il variare della recall in funzione del numero di utenti nel dominio ausiliario ed è corredato da una tabella che mostra invece i valori puntuali ottenuti dalle diverse metriche in caso di raccomandazione top-N con N pari a 20.

In Figura 7.1 è mostrato come cambia la recall delle raccomandazioni compiute sul dominio target al variare del numero di utenti presenti nel dominio ausiliario, o training set. In particolare in Figura 7.1a sono mostrati i valori di recall per un numero di utenti pari a 0.01%, 0.1%, 1% e 10% della URM originale di Netflix. Poiché la matrice di partenza è composta dai rating di quasi 500000 utenti, i valori di recall riportati per ogni metodo corrispondono ai casi in cui il training set è composto rispettivamente da circa 50, 500, 5000 e infine 50000 utenti. In particolare la scelta di impiegare una scala logaritmica ha permesso di evidenziare il comportamento dei diversi algoritmi per valori molto piccoli e ravvicinati del numero di utenti selezionati. In Figura 7.1b sono invece riportati i valori per una diversa percentuale di utenti pari a 20%, 40%, 60% ed 80% a cui corrispondono un numero di individui tra 100000 e 400000.

In Tabella 7.1 e 7.2 sono invece riportati i valori puntuali per le tre metriche utilizzate: recall, fallout e AUC. I valori riportati all'interno delle tabelle si riferiscono alle grandezze valutate per raccomandazioni top-N dove N è pari a 20. Ciascun valore è seguito dalla variazione osservata per la medesima metrica rispetto al passo precedente. La prima tabella mostra i valori per un numero di utenti compresi tra 0.01% e 10% della URM originale, pari a un numero compreso tra 50 e 50000, mentre la seconda dal 20% al 80%, con valori assoluti tra 100000 e 400000 utenti. Ciascuna tabella principale è suddivisa in quattro parti che riportano gli esiti degli esperimenti condotti utilizzando i diversi partizionamenti della URM di Netflix. I valori del dominio target, non riportati, sono pari alla differenza insiemistica tra la URM e il training set. Ad esempio ad un training set pari al 10% è associato un dominio target, o test set, pari al 90%. Osservando ad esempio la Tabella 7.2, il metodo PureSVD al primo passo, con training set pari a 0.01% della URM originale, mostra una recall pari a 0.5725 mentre al secondo passo essa vale 0.7045 con uno scarto di 0.132. Da notare che dati i bassi valori, le quantità riportate per gli scarti sono moltiplicate per 100 per semplificarne la lettura. Infine la colonna AUC di ciascuna tabella indica l'area sottesa alla curva ottenuta ponendo il fallout sull'asse delle ordinate e la recall su quella delle ascisse.



(a) Utenti da 50 a 50000



(b) Utenti da 100000 a 400000

Figura 7.1: Grafico delle recall ottenute con il partizionamento dei dataset per dimensione, considerando un numero di utenti da 50 a 50000 (a) e da 100000 a 400000 (b).

	training set = 0.01% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.029		**0.0285**		0.6565	
TopRated	0.5195		0.368		0.5976	
Cosine	0.5425		0.3		0.6479	
Pearson	0.343		0.129		0.6461	
PureSVD	**0.5725**		0.3175		0.6522	
AsySVD	0.348		0.1		**0.6571**	

	training set = 0.1% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.006	-2.3	**0.0077**	2.08	0.734	7.75
TopRated	0.533	1.35	0.3895	-2.15	0.5996	0.2
Cosine	0.669	12.65	0.3875	-8.75	0.6985	5.06
Pearson	0.238	-10.5	0.0285	**10.05**	0.7519	10.58
PureSVD	**0.7045**	**13.2**	0.3625	-4.5	0.7243	7.21
AsySVD	0.3475	-0.05	0.0765	2.35	**0.7701**	**11.3**

	training set = 1% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.007	0.1	**0.0053**	0.245	0.8364	10.24
TopRated	0.537	0.4	0.385	0.45	0.6069	0.73
Cosine	0.7125	4.35	0.393	-0.55	0.7207	2.22
Pearson	0.206	-3.2	0.0145	1.4	0.8588	**10.69**
PureSVD	**0.753**	**4.85**	0.3765	-1.4	0.7542	2.99
AsySVD	0.3305	-1.7	0.0313	**4.525**	**0.8752**	10.51

	training set = 10% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.152	**14.5**	0.0247	-1.945	0.7795	-5.686
TopRated	0.5285	-0.85	0.3773	0.77	0.6056	-0.13
Cosine	0.727	1.45	0.3763	**1.67**	0.7299	0.92
Pearson	0.236	3	**0.012**	0.25	0.8701	1.13
PureSVD	**0.77**	1.7	0.378	-0.15	0.7537	-0.05
AsySVD	0.345	1.45	0.015	1.625	**0.9186**	**4.34**

Tabella 7.1: Risultati ottenuti per test su dataset artificiali nel caso di partizionamento basato sul numero degli utenti. Viene indicata per ogni gruppo di test la percentuale di utenti della URM originale utilizzati nel training set mentre il test set è costituito dalla parte restante di URM. Per ogni metrica ed ogni algoritmo è indicato il valore ottenuto e la variazione riscontrata con il passo precedente.

	training set = 20% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.1629		0.029		0.7822	
TopRated	0.5517		0.378		0.6089	
Cosine	0.7271		0.391		0.726	
Pearson	0.2089		**0.013**		0.8571	
PureSVD	**0.7611**		0.375		0.7587	
AsySVD	0.3653		0.0135		**0.9207**	

	training set = 40% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.1635	0.06	0.028	0.1	0.7864	0.42
TopRated	0.5525	0.08	0.376	**0.2**	0.6153	0.64
Cosine	0.736	0.89	0.392	-0.1	0.7337	**0.77**
Pearson	0.2875	**7.86**	0.018	-0.5	0.8439	-1.32
PureSVD	**0.784**	2.29	0.3875	-1.25	0.7687	1
AsySVD	0.4085	4.32	**0.014**	-0.05	**0.9264**	0.566

	training set = 60% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.188	**2.45**	0.03	-0.2	0.7767	-0.97
TopRated	0.5375	-1.5	0.3878	-1.18	0.6015	-1.38
Cosine	0.725	-1.1	0.4058	-1.38	0.7268	-0.69
Pearson	0.2935	0.6	0.026	-0.8	0.8296	-1.43
PureSVD	**0.7885**	0.45	0.3863	**0.12**	0.7603	-0.84
AsySVD	0.4105	0.2	**0.0165**	-0.25	**0.9253**	-0.11

	training set = 80% utenti					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.1675	-2.05	0.0295	0.05	0.7812	**0.45**
TopRated	0.5365	-0.1	0.388	-0.02	0.603	0.15
Cosine	0.721	-0.4	0.3975	**0.38**	0.7165	-1.03
Pearson	0.304	**1.05**	0.0255	0.05	0.824	-0.56
PureSVD	**0.772**	-1.65	0.388	-0.17	0.7563	-0.4
AsySVD	0.4125	0.2	**0.013**	0.35	**0.9289**	0.36

Tabella 7.2: continua Tabella 7.1

Discussione dei risultati

Si considerino la Figura 7.1a, che mostra i soli valori di recall, e la Tabella 7.1, che include invece tutte le metriche. Concentriamoci inizialmente sulla recall. Il primo dato importante che emerge è la superiorità dell'algoritmo PureSVD per qualunque variazione del numero di utenti. Quando la percentuale è minima, pari a 0.01% degli utenti originali di Netflix, il valore di recall è già buono (0.5725) indicando che le poche informazioni a disposizione sono già sufficienti per compiere raccomandazioni sulla restante parte del dataset. Questo fenomeno può essere spiegato osservando l'andamento dell'algoritmo TopRated che già in partenza sembra aver raggiunto il suo massimo. Questo comportamento mostra come già 50 utenti siano sufficienti per identificare gli item più popolari all'interno del dataset generando le raccomandazioni più semplici da calcolare.

I metodi Cosine e Pearson, che valutano entrambi la similarità tra item ma utilizzando tecniche differenti, presentano comportamenti molto diversi. Il primo evidenzia un valore di recall iniziale già molto buono, segno che gli item più popolari hanno un peso maggiore in questo algoritmo, ed inoltre migliora all'aumentare del numero di utenti. Il secondo invece mostra indifferenza ai best seller e non trae giovamento dall'aumento delle dimensioni del training set. Osservando tuttavia i valori di AUC per questo algoritmo se ne osserverà il costante aumento segno della riduzione del fallout.

AsymmetricSVD mostra un comportamento stabile per quanto riguarda la recall anche se l'aumento del numero di utenti produce una riduzione del fallout che si evidenzia nella tabella osservando il valore della metrica AUC. I valori così elevati rispetto agli altri metodi significano che a parità di fallout l'algoritmo AsymmetricSVD esibisce una recall superiore.

L'algoritmo MovieAVG presenta valori di fallout molto bassi accompagnati però da recall altrettanto ridotte, il che rende questo metodo di scarso interesse.

Si osservino ora la Figura 7.1b e la Tabella 7.2, esse mostrano i valori delle metriche per un numero di utenti sensibilmente superiore al caso precedente. Tutti gli algoritmi, fatto salvo Pearson, mostrano un comportamento piatto, segno che hanno già raggiunto il massimo della loro qualità e l'aumento di utenti non aggiunge alcuna informazione utile alla creazione del modello. L'algoritmo Pearson mostra invece un andamento crescente ma sempre modesto a cui però segue l'attestazione del valore di AUC a valori ragguardevoli e pari a circa 0.83, segno che la qualità di questo metodo è da ricercarsi più nel basso fallout che nell'elevata recall.

Da questa prima analisi emerge come l'algoritmo migliore in termini di recall sia PureSVD che è in grado di sfruttare la presenza di item popolari in caso

di un numero esiguo di utenti su cui calcolare il modello. Per contro il metodo AsymmetricSVD esibisce la miglior AUC per qualunque dimensione del dataset mostrando una recall scarsamente dipendente dal numero di utenti, che influenzano invece la qualità del fallout. Si noti infine che i due metodi migliori individuati in questa fase di test fanno uso della decomposizione SVD a riprova della bontà che riescono a raggiungere gli approcci basati su fattori latenti. Si può concludere quindi che per questi metodi anche un numero esiguo di utenti è sufficiente per generare delle buone raccomandazioni.

7.2 Analisi sulla variazione della densità

In questo paragrafo sono riportati ed analizzati i risultati ottenuti dai test preliminari compiuti variando la densità del training set per valutare l'impatto sulla qualità delle raccomandazioni.

In Figura 7.2 è mostrato come cambia la recall delle raccomandazioni compiute sul dominio target al variare della densità del dataset associato al dominio ausiliario. Il dominio ausiliario è stato creato utilizzando metà utenti della URM originale di Netflix e la densità è stata fatta variare prendendo un numero di rating variabili tra un minimo dello 0.01% e un massimo pari al 10%. Il valore limite del 100% è stato ottenuto dalla Figura 7.1b prendendo il punto pari al 50% degli utenti, per cui metà degli utenti costituiscono il training set e l'altra metà il test set.

Come per le tabelle precedenti, anche in Tabella 7.3 sono riportati i valori puntuali di recall, fallout e AUC per una raccomandazione top-N dove N è pari a 20. Accanto ad ogni valore è inoltre riportata la variazione rispetto al passo precedente. La tabella mostra i valori della densità del training set compresi tra 0.01% e 10%, rispetto alla URM originale presenta una densità pari a 1.18%. I risultati di ognuna delle prove sono riportati in una delle quattro porzioni della tabella. Il test set invece è rimasto invariato sia come densità che come dimensione, pari al 50% della URM originale. Ad esempio il metodo PureSVD al primo passo, con training set pari a 0.01% della URM originale, mostra una recall pari a 0.5725 mentre al secondo passo essa vale 0.7045 con uno scarto di 0.132. Da notare che dati i bassi valori, le quantità riportate per gli scarti sono moltiplicate per 100 per semplificarne la lettura. Infine la colonna AUC di ciascuna tabella indica l'area sottesa alla curva ottenuta ponendo il fallout sull'asse delle ordinate e la recall su quella delle ascisse.

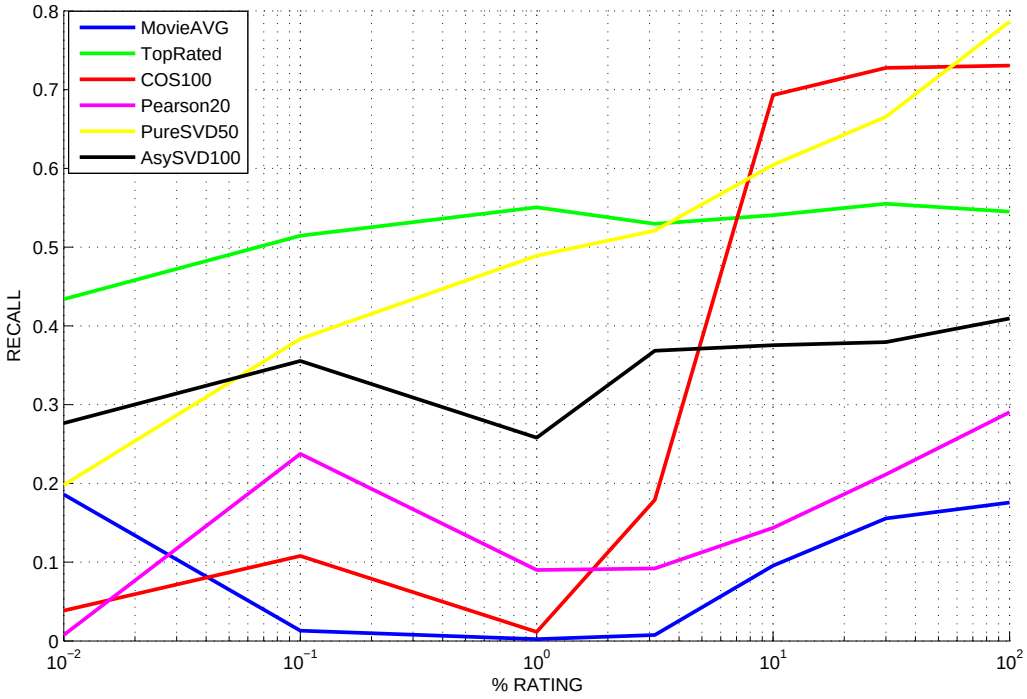


Figura 7.2: Grafico delle recall ottenute con il partizionamento dei dataset basato sulla densità, considerando valori compresi tra 0.01% e 100% del training set (pari alla metà della URM originale)

	training set = 0.01% rating					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.186		0.0875		0.6167	
TopRated	**0.434**		0.2974		0.5869	
Cosine	0.0385		0.0295		0.508	
Pearson	0.0075		**0.007**		0.5117	
PureSVD	0.198		0.1544		0.4942	
AsySVD	0.2765		0.1159		**0.628**	

	training set = 0.1% rating					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.013	-17.3	**0.014**	**7.35**	0.6919	7.52
TopRated	**0.5145**	8.05	0.3688	-7.14	0.6005	1.36
Cosine	0.108	6.95	0.0675	-3.801	0.5864	7.84
Pearson	0.2375	**23**	0.1289	-12.19	0.587	7.53
PureSVD	0.3835	18.55	0.2474	-9.3	0.6039	**10.97**
AsySVD	0.3555	7.9	0.1004	1.55	**0.702**	7.4

	training set = 1% rating					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.0022	-1.08	**0.0039**	1.01	0.7539	6.2
TopRated	**0.5505**	3.6	0.3753	-0.65	0.6091	0.86
Cosine	0.0115	-9.65	0.0155	5.2	0.4516	-13.48
Pearson	0.09	-14.75	0.02	**10.89**	0.7013	11.43
PureSVD	0.489	**10.55**	0.3303	-8.29	0.5995	-0.44
AsySVD	0.258	-9.75	0.0505	4.99	**0.7698**	6.78

	training set = 10% rating					
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100
MovieAVG	0.0955	9.33	0.0115	-0.76	0.7799	2.6
TopRated	0.5405	-1	0.3738	0.15	0.6091	0
Cosine	**0.693**	**68.15**	0.3928	-37.73	0.7073	**25.57**
Pearson	0.1435	5.35	**0.0095**	1.05	0.8195	11.82
PureSVD	0.6045	11.55	0.3443	-1.4	0.6766	7.71
AsySVD	0.3755	11.75	0.031	**1.95**	**0.8807**	11.09

Tabella 7.3: Risultati ottenuti per test su dataset artificiali nel caso di partizionamento basato sulla variazione della densità. Viene indicata per ogni gruppo di test la percentuale di rating della URM originale utilizzati nel training set mentre il test set è rimasto invariato e pari a metà della URM. Per ogni metrica ed ogni algoritmo è indicato il valore ottenuto e la variazione riscontrata con il passo precedente.

Discussione dei risultati

Si considerino la Figura 7.2, che mostra i valori della recall al variare della percentuale di rating, e la Tabella 7.3, che riporta i valori per ciascuna metrica. Da una prima analisi emerge, come nei test precedenti, che la presenza di item popolari consente all'algoritmo TopRated di ottenere buoni risultati anche per valori estremamenti ridotti di densità. Questo fenomeno è indice di come gli item più popolari siano già evidenti anche a densità minime.

Il profilo irregolare che mostrano i diversi metodi è indice della mancata ottimizzazione in questa fase preliminare di test e dell'utilizzo dei valori di default di ciascun algoritmo. Questo produce risultati variabili poiché per talune densità i parametri risulteranno migliori che per altre.

I metodi Cosine e Pearson, che entrambi valutano la similarità tra i diversi oggetti con strategie diverse, mostrano inizialmente comportamenti simili. Quando però il numero di rating passa da 1% a 10% è possibile osservare un significativo miglioramento dell'algoritmo Cosine, pari al 68%, mentre Pearson non ne trova grande giovamento. Inoltre il metodo Cosine presenta una recall superiore agli altri metodi finché il numero di rating si mantiene inferiore al 70% di quelli presenti nella URM originale.

Il metodo PureSVD esibisce una qualità della recall in costante aumento al crescere del numero di rating segno di come la densità del training set sia un parametro importante per questo algoritmo.

AsymmetricSVD mostra nuovamente, come nel test precedente, valori di recall stabili associati a fallout in costante diminuzione. Questa combinazione di fattori produce valori di AUC superiori ad ogni altra soluzione.

Questi test hanno mostrato come gli algoritmi PureSVD e Cosine siano i migliori in termini di recall mentre AsymmetricSVD presenti la miglior AUC. Considerando che il dataset Netflix da cui sono originati i dati presenta una densità del 1.18%, emerge come anche per valori molto ridotti della densità questi metodi siano in grado di garantire raccomandazioni già di ottima qualità.

Capitolo 8

Analisi su dataset reali

In questo capitolo vengono presentati e discussi i risultati ottenuti dai test compiuti sui dataset reali MovieLens e Yahoo!. Gli algoritmi utilizzati per questa fase di test sono stati ottimizzati, come discusso nel Paragrafo 4.3, così che ciascuno manifesti la migliore qualità di raccomandazione possibile. I test presentati in questo capitolo mostrano il comportamento del porting tra dataset reali così come i risultati ottenuti dai metodi content-based, riportati come riferimento. Non vengono discussi gli esiti dei metodi ibridi poiché non hanno conseguito risultati rilevanti. In particolare gli algoritmi collaborativi presentati sono:

- MovieAVG: Movie Average, non personalizzato.
- TopRated: Top Popular, non personalizzato.
- COS200: item-based collaborativo, Cosine con parametro k-NN = 200.
- Pearson20: item-based collaborativo, Pearson con parametro k-NN = 20.
- Pearson50: item-based collaborativo, Pearson con parametro k-NN = 50.
- PureSVD50: item-based collaborativo, PureSVD con 50 fattori latenti.
- AsySVD50: item-based collaborativo, Asymmetric SVD con 50 fattori latenti.

Si noti come i due algoritmi Pearson con parametro differente vengano usati in presenza di due domini ausiliari differenti. Quando il training set è costituito dal dataset Yahoo! il parametro ottimale è pari a 20 mentre per il caso MovieLens è 50.

Per quanto concerne i test con metodi content-based, quelli utilizzati sono:

- CosCon: item-based, Cosine Content.

- DirectContent100: item-based, Direct Content con parametro k-NN = 100.
- NaïveBayes: classificatore bayesiano.
- LsaCosine50ls250: item-based, LSA con 250 fattori latenti e parametro k-NN = 50.
- LsaCosine50ls300: item-based, LSA con 300 fattori latenti e parametro k-NN = 50.

Gli algoritmi LSA presentano due valori differenti per la latent size, il primo è ottimale per i test su dataset Yahoo! mentre il secondo è più indicato per quelli su dataset MovieLens.

Vengono di seguito esposti i risultati ottenuti mentre la loro discussione è affrontata nel Paragrafo 8.2.

8.1 Presentazione dei risultati

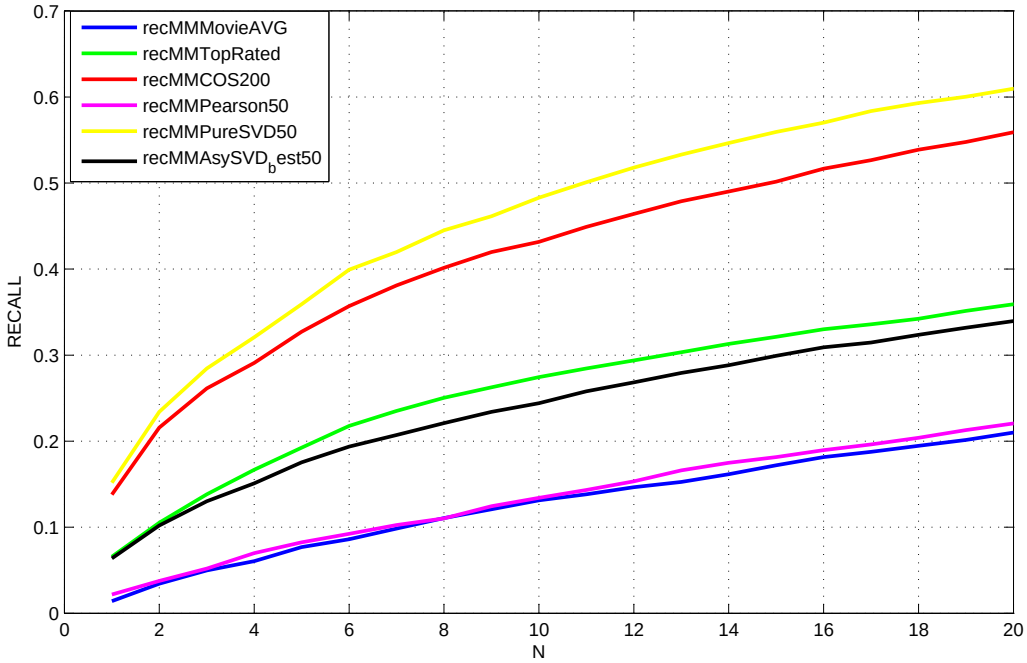
In questo paragrafo vengono riportati i risultati ottenuti dai metodi collaborativi e content-based. Ciascun gruppo di algoritmi è presentato con grafici, che indicano i valori della recall al variare del parametro N tra 1 e 20 in raccomandazioni *top-N*. Sono presenti inoltre tabelle che riportano i valori puntuali assunti dalle diverse metriche per N=20.

Metodi collaborativi

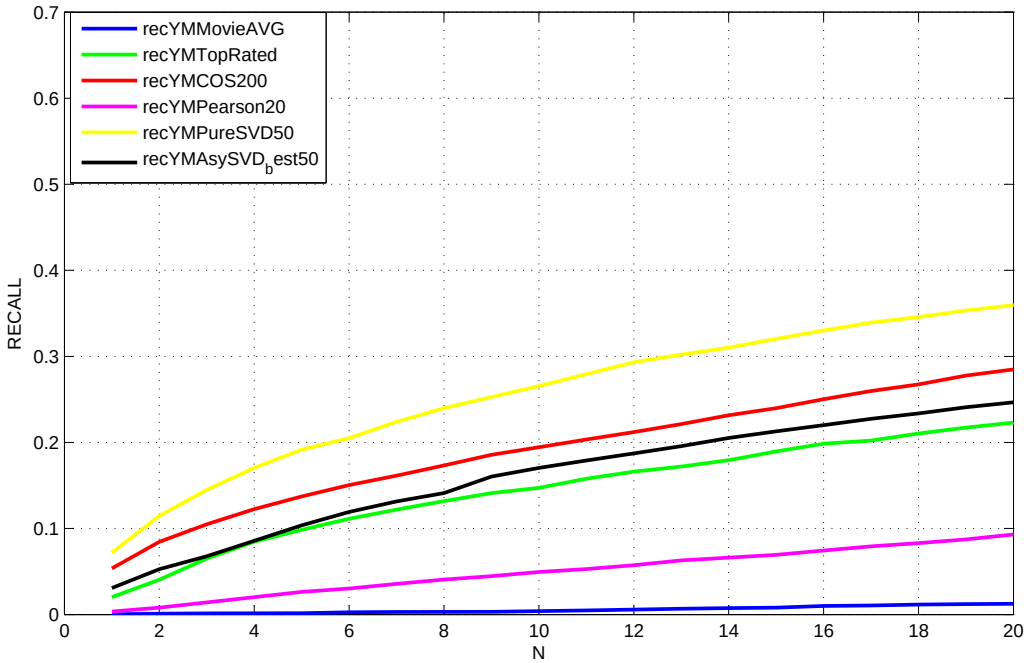
In Figura 8.1 è mostrato come cambia la recall al variare del dominio ausiliario utilizzato per apprendere il modello. In particolare in Figura 8.1a è riportato il caso ideale in cui dominio target e ausiliario coincidono mentre in Figura 8.1b si evidenzia l'effetto del porting del modello appreso sul dataset Yahoo! e testato quindi su quell MovieLens.

In Figura 8.2 si propone un caso analogo al precedente, dove vengono riportati sia i risultati ottenuti nel caso ideale che quelli ricavati in presenza di porting. In questo caso si tratta di un modello appreso dal dataset MovieLens e testato su quello Yahoo!, come mostrato in Figura 8.2b.

La Tabella 8.1 riporta i valori puntuali per le tre metriche utilizzate: recall, fallout e AUC. I valori indicati all'interno di ciascuna sottotabella si riferiscono alle metriche valutate in raccomandazioni top-N con N = 20. Ciascun valore è seguito dalla variazione osservata per la medesima metrica rispetto al caso di porting ideale, cioè quando i domini ausiliario e target coincidono. La tabella principale è costituita da quattro sottotabelle, dove ciascuna mostra i risultati per

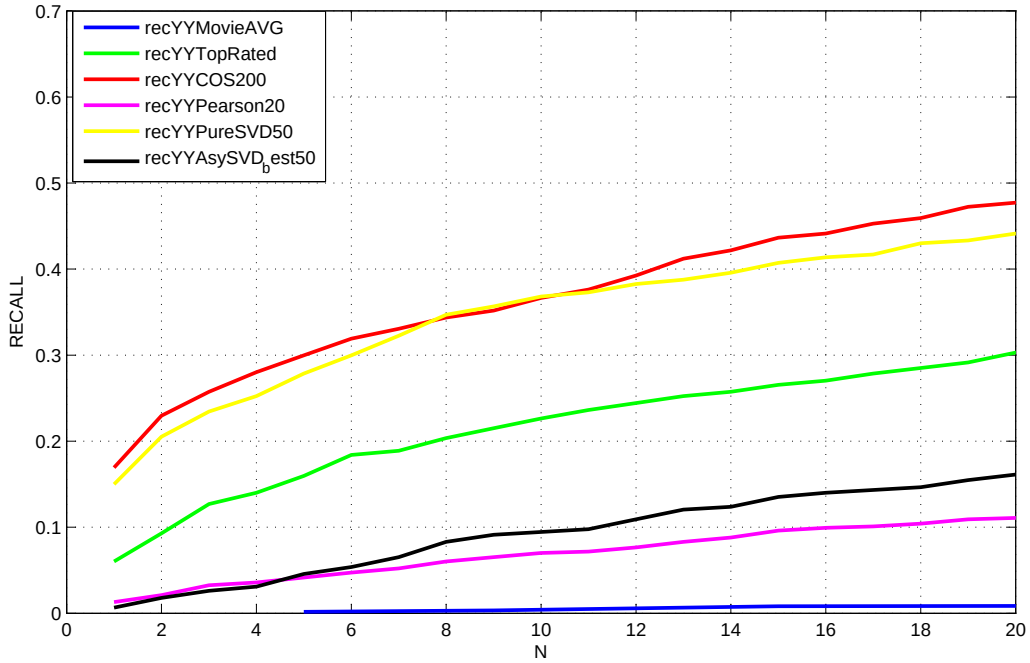


(a) Dominio ausiliario MovieLens (caso *2* della Tabella 4.1)

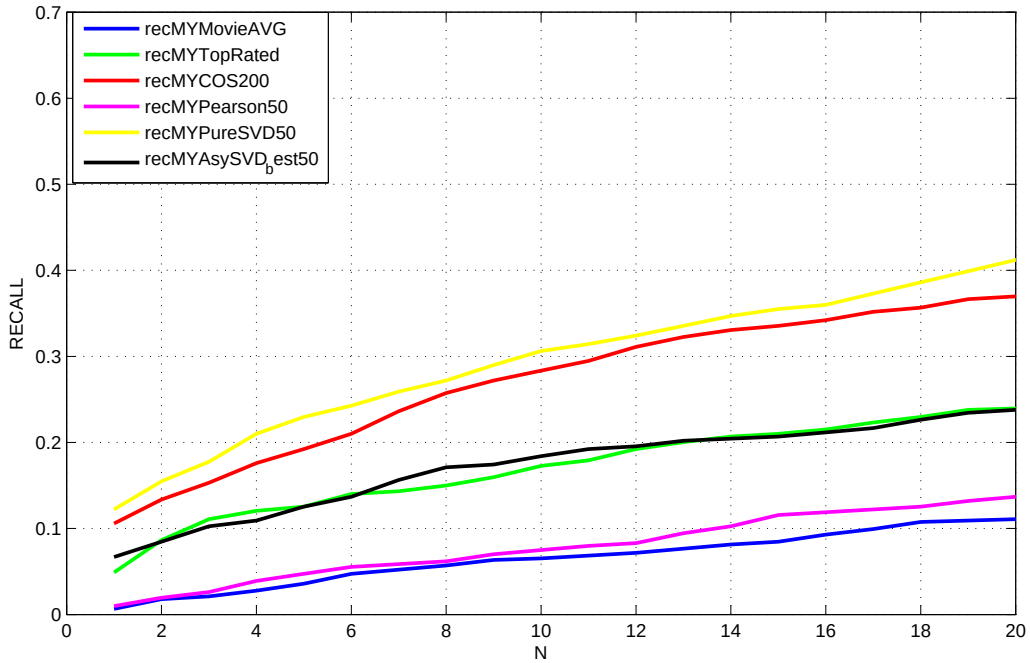


(b) Dominio ausiliario Yahoo! (caso *3* della Tabella 4.1)

Figura 8.1: Recall dei metodi collaborativi testati sul dominio target MovieLens



(a) Dominio ausiliario Yahoo! (caso *4* della Tabella 4.1)



(b) Dominio ausiliario MovieLens (caso *1* della Tabella 4.1)

Figura 8.2: Recall dei metodi collaborativi testati sul dominio target Yahoo!

una diversa combinazione di dataset di training e di test. Le sottotabelle appaiate per righe utilizzano il medesimo dominio target mentre quelle disposte una sopra l'altra si riferiscono a identici domini ausiliari. L'analisi tra il caso ideale e il porting può quindi essere fatta osservando le sottotabelle disposte una accanto all'altra. Ad esempio quelle riportate sulla prima riga indicano test compiuti sul dominio target MovieLens.

Metodi content-based

In Figura 8.3 sono riportati i valori ottenuti dai test effettuati con algoritmi content-based sui due diversi dataset MovieLens e Yahoo!. Poiché per i metodi content-based non ha senso effettuare il porting, come discusso nel Paragrafo 4.2, il modello è stato appreso su un dataset comune denominato *best* e i test sono stati compiuti sui due diversi dataset. Il primo grafico, rappresentato in Figura 8.3a, mostra i valori della recall di ciascun algoritmo al variare del parametro N, in raccomandazioni top-N, per il dataset MovieLens. Il secondo grafico riporta invece le medesime grandezze riferite ai test sul dataset Yahoo!.

In Tabella 8.2 vengono riportati i valori di recall, fallout e AUC associati ai test content-based effettuati sui dataset MovieLens e Yahoo!. La tabella principale è divisa in due sottotabelle, la prima a sinistra mostra la qualità delle raccomandazioni compiute sul dataset MovieLens mentre l'altra riporta i valori relativi alle raccomandazioni sul dataset Yahoo!. Per entrambi i test si è utilizzato il medesimo dataset di training denominato *best* e presentato nel Paragrafo 4.2.

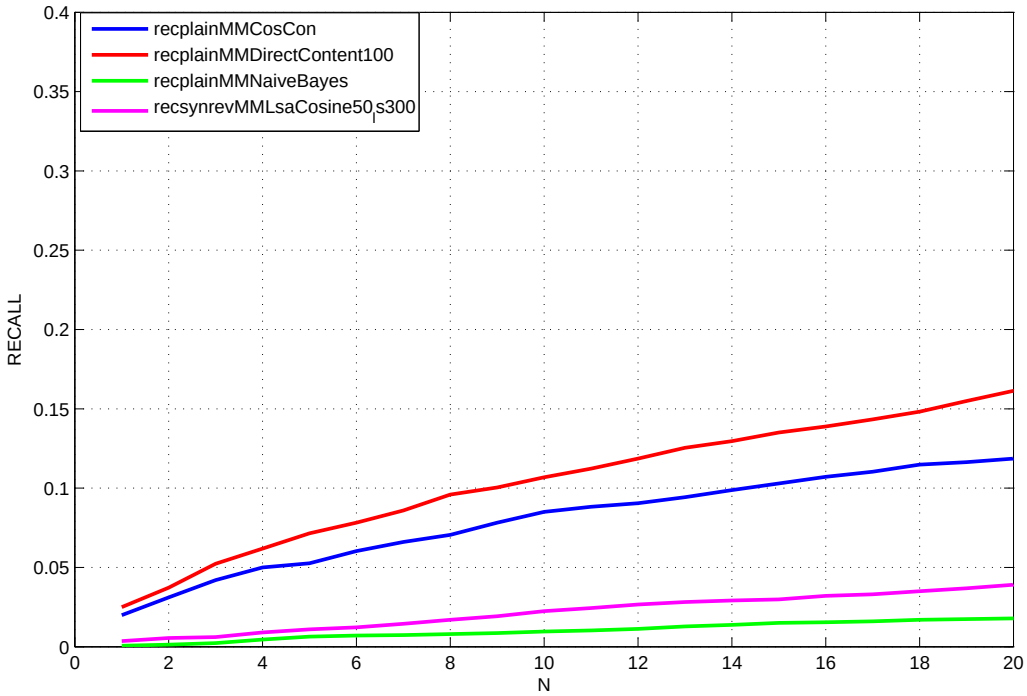
8.2 Discussione dei risultati

Si consideri la Figura 8.1, che mostra il caso di porting dal dominio ausiliario Yahoo! a quello target MovieLens. É evidente come tutti i metodi ottengano valori di recall inferiori quando il modello è stato appreso su un dataset differente (vedi Figura 8.1b) rispetto al caso in cui training e test set coincidano (vedi Figura 8.1b). I risultati sono ancora più evidenti analizzando la Tabella 8.1. Osservando i valori di recall e AUC si notano infatti forti peggioramenti per tutti gli algoritmi, anche se il metodo PureSVD rimane il migliore per la metrica recall mentre AsymmetricSVD presenta i valori più elevati di AUC in entrambi i casi.

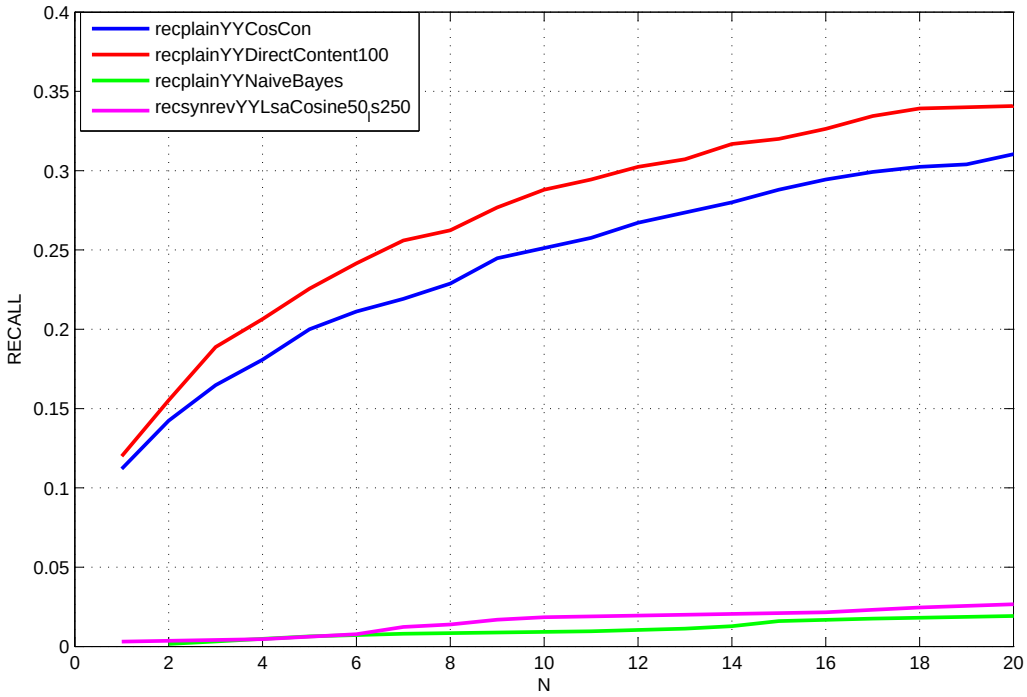
A fronte di questi risultati è lecito domandarsi se un approccio content-based sarebbe stato più indicato del tentativo di porting del modello. Analizzando la Figura 8.3a e la Tabella 8.2, appare come nessun algoritmo riesca ad ottenere una recall superiore a quella raggiunta dal metodo collaborativo PureSVD. Per quanto riguarda la metrica AUC l'algoritmo collaborativo AsymmetricSVD risulta

	dominio ausiliario: MovieLens						dominio ausiliario: Yahoo!						
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100	
MovieAVG	0.2101		0.0118		0.9019		0.0125	-19.76	**0.0021**	0.97	0.8135	**-8.84**	**target: MovieLens**
TopRated	0.3592		0.0604		0.7729		0.2232	-13.6	0.0578	0.26	0.6126	-16.03	
Cosine	0.5590		0.1472		0.8085		0.2848	-27.42	0.0696	**7.76**	0.7158	-9.27	
Pearson	0.2207		0.0085		0.9312		0.093	-12.77	0.0092	-0.07	0.7528	-17.84	
PureSVD	**0.6097**		0.1577		0.8059		**0.3595**	-25.02	0.0894	6.83	0.7149	-9.1	
AsySVD	0.3396		**0.0063**		**0.9461**		0.2466	**-9.3**	0.0263	-2	**0.8422**	-10.39	
	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100	R@20	Δ x100	F@20	Δ x100	AUC	Δ x100	
MovieAVG	0.1107	**10.21**	0	0	0.7268	-2.23	0.0086		0		0.7491		**target: Yahoo!**
TopRated	0.2394	-6.35	0	14	0.6912	9.95	0.3029		0.14		0.5917		
Cosine	0.3697	-10.75	0.0636	**15.31**	0.7318	10.28	**0.4772**		0.2167		0.629		
Pearson	0.1368	2.61	0	0	0.5660	2.92	0.1107		0		0.5368		
PureSVD	**0.4121**	-2.93	0.0844	12.56	0.7525	**12.18**	0.4414		0.2125		0.6307		
AsySVD	0.2378	7.66	0	7.35	**0.8300**	7.43	0.1612		0.0735		**0.7557**		

Tabella 8.1: Risultati ottenuti dai metodi collaborativi su dataset reali (MovieLens e Yahoo!). Ognuna delle quattro sottotabelle mostra i risultati per un diverso dataset di training e di test. L'analisi tra il caso ideale e il porting può essere fatta osservando le tabelle disposte una accanto all'altra. Ad esempio le prime due indicano test compiuti sul medesimo dominio target MovieLens. Per ogni metrica ed ogni algoritmo è indicato il valore ottenuto e la variazione riscontrata rispetto al caso ideale.



(a) Test su dataset MovieLens (caso *b* della Tabella 4.2)



(b) Test su dataset Yahoo! (caso *a* della Tabella 4.2)

Figura 8.3: Recall ottenute dai metodi content-based sui dataset reali

	target: MovieLens			target: Yahoo!		
	R@20	F@20	AUC	R@20	F@20	AUC
Naïve	0.0179	0.0255	0.4495	0.0192	0	0.5368
Cosine	0.1187	0.0828	0.5056	0.3104	0.2143	0.5224
Direct	**0.1613**	0.0683	0.5646	**0.3408**	0.2083	**0.5532**
LSA	0.0391	**0.0025**	**0.7364**	0.0267	0.059	0.4503

Tabella 8.2: Risultati ottenuti dai metodi content-based sui dataset reali MovieLens e Yahoo!. Il training set, denominato best, è unico mentre ciascun dataset viene usato come test set.

anch'esso superiore alle alternative content-based. Da questi risultati emerge come anche nel caso in cui il porting degradi notevolmente la qualità delle raccomandazioni rispetto al caso ideale esso resti comunque la scelta migliore rispetto alle tecniche content-based.

Si consideri ora la Figura 8.2, che mostra invece il porting dal dominio ausiliario MovieLens a quello target Yahoo!. I grafici evidenziano cali meno significativi della recall rispetto al test precedente e si intravede qualche inatteso miglioramento. Analizzando la Tabella 8.1 appare anche evidente come il porting consenta di ridurre notevolmente il fallout così da condurre ad un aumento della AUC per quasi tutti i metodi. In particolare l'algoritmo PureSVD si conferma il migliore in termini di recall mentre AsymmetricSVD presenta i valori di AUC più elevati. Risulta pertanto che le raccomandazioni compiute utilizzando il porting del modello sono persino migliori rispetto al caso ideale. Un breve sguardo alla Figura 8.3b e alla Tabella 8.2 mostrano come i metodi collaborativi ottengano nuovamente prestazioni inferiori rispetto a quelli collaborativi in presenza di porting.

I test effettuati sui dataset reali hanno mostrato come il porting sia una tecnica efficace per supplire alla mancanza di informazioni del dominio target. Apprendendo un modello sul dominio ausiliario consente comunque di ottenere risultati che mostrano qualità superiori rispetto ai metodi content-based. Infine è possibile che il porting del modello ottenga addirittura risultati migliori persino del caso ideale in cui i due domini coincidano. Questa evenienza è spiegabile con la differenza di densità tra i due dataset, MovieLens e Yahoo!, dove il primo presenta una densità superiore di 15 volte rispetto al secondo. L'elevato numero di rating nel dataset MovieLens consente quindi la creazione di un modello migliore, che in fase di raccomandazione ottiene prestazioni superiori rispetto a quello costruito

sul dataset Yahoo!.

Capitolo 9

Conclusioni e sviluppi futuri

Al termine di questo lavoro di tesi è opportuno trarre alcune considerazioni generali sui risultati ottenuti e sui possibili scenari futuri che potrebbero seguire a questo studio. Lo scopo principale di questa tesi era quello di valutare la realizzazione di sistemi di raccomandazione per contesti televisivi basati su canali di distribuzione uniderezionali. Alla luce delle idee proposte e dei risultati ottenuti è evidente come tutte le principali famiglie di algoritmi possano essere adattate per funzionare in questo nuovo contesto operativo. Se per alcuni approcci le modifiche richieste risultano moderate (non personalizzati, demografici e content-based), per altri lo sforzo richiesto è decisamente maggiore; è il caso degli algoritmi collaborativi per cui è stata introdotta e valutata l'idea innovativa denominata *porting* del modello.

I risultati ottenuti in fase di testing mostrano come il porting del modello da un dominio ausiliario (con canale bidirezionale) sia una valida idea per supplire alla mancanza di informazioni nel dominio target (con canale unidirezionale), confermando la congettura secondo la quale se si osserva un certo fenomeno sul primo dataset è molto probabile che questo si verifichi anche nel secondo. L'implicazione che questa affermazione pone nel contesto televisivo è al contempo intuitiva e forte, ovvero che gli utenti di piattaforme diverse mostreranno in media gli stessi comportamenti.

La fase di testing preliminare condotta su Netflix ha consentito di valutare come la densità dei rating e il numero di utenti del dominio ausiliario, o training set, influiscano sulla bontà delle raccomandazioni. Superata una soglia iniziale, le due grandezze contribuiscono in misura ridotta soprattutto per alcuni algoritmi che risultano particolarmente robusti a variazioni di questi parametri. Per quanto concerne la densità dei rating, gli algoritmi hanno presentato miglioramenti fino a raggiungere valori di regime in corrispondenza di una densità del training set pari a circa 0.12%, corrispondente al 10% della densità originale. Per quanto

riguarda il numero di utenti è apparso come circa 5000 individui, pari all'1% di quelli presenti nella URM originale, siano sufficienti per raggiungere una qualità di regime.

Un'ulteriore fase di testing è stata compiuta su due dataset reali, MovieLens e Yahoo!, che condividono parte degli elementi, utilizzandoli alternativamente come dominio target e dominio ausiliario. Da questi esperimenti è emerso un riscontro al fenomeno sopracitato di dipendenza della qualità dalla densità del training set, mostrando come il porting possa essere impiegato con maggior successo su alcuni dataset rispetto ad altri. Si è così potuto verificare come il modello costruito su un dominio ausiliario con una densità elevata (MovieLens) possa essere utilizzato con successo per compiere raccomandazioni su un dominio target. Lo stesso esperimento di porting, ma partendo da un dominio ausiliario sparso (Yahoo!), ha evidenziato prestazioni meno confortanti che rendono comunque la soluzione con porting superiore a quelle content-based.

Per alcuni algoritmi si è inoltre verificato un fenomeno ancora più interessante, per cui il porting da un dominio ausiliario denso ad uno target sparso ha portato a dei miglioramenti nella qualità delle raccomandazioni. Questi risultati avvalorano ancor più la tesi che entrambi i dataset, seppur raccolti da fornitori differenti, descrivano il medesimo fenomeno.

Infine gli algoritmi che meglio si prestano al porting sono PureSVD ed AsymmetricSVD. Il primo presenta le migliori prestazioni in termini di recall mentre il secondo è il migliore in valutazioni che considerano AUC. Si noti che entrambi i metodi fanno uso della decomposizione SVD che mostra una buona capacità di astrarre le caratteristiche del dominio ausiliario producendo un modello in grado di generare buone raccomandazioni sul dominio target.

L'utilizzo di sistemi di raccomandazione su canali di distribuzione unidirezionali modifica significativamente il livello della privacy degli utenti rispetto ai sistemi tradizionali. Infatti l'impossibilità per il set-top box di inviare le informazioni al service provider garantisce un completo anonimato. Anche nell'eventualità di porting del modello, le informazioni trasferite da un provider ad uno differente sarebbero del tutto prive di informazioni utili ad identificare gli utenti.

Per avere ulteriore conferma dei risultati fin qui presentati, le soluzioni ideate in questo lavoro di tesi potrebbero essere sperimentate su sistemi reali. Per quelli basati su meccanismi di pay-per-view sarebbe interessante valutare se questo tipo di raccomandazioni personalizzate abbiano un reale impatto in termini di revenue, spingendo per un'introduzione del porting su larga scala. Invece per le piattaforme televisive premium, che includono nell'offerta un insieme di contenuti già acquistati, sarebbe importante valutare l'impatto sulla customer satisfaction

e, più a lungo termine, sulla customer retention.

Bibliografia

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

[2] Jae-Wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems: help or harm? *International World Wide Web Conference*, page 11, 2007.

[3] F Airoldi, P Cremonesi, and R Turrin. Hybrid algorithms for recommending new items in personal TV. In *2nd Workshop on Future Television at EuroITV 2011: making television personal and social*, 2011.

[4] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

[5] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[6] Nicola Barbieri and Giuseppe Manco. An Analysis of Probabilistic Methods for Top-N Recommendation in Collaborative Filtering. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6911 of *Lecture Notes in Computer Science*, pages 172–187. Springer Berlin / Heidelberg, 2011.

[7] Robert M. Bell and Yehuda Koren. *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights*. IEEE, October 2007.

[8] Edward Louis Bernays. *Propaganda*. Horace Liveright, New York, New York, USA, 1928.

[9] Daniel Billsus and Michael J. Pazzani. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, February 2000.

[10] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[11] Robin Burke. Hybrid Web Recommender Systems. *The Adaptive Web*, 4321:377 – 408, 2007.

[12] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. You Might Also Like :âĂİ Privacy Risks of Collaborative Filtering. *Computer*, pages 231–246, 2011.

[13] Elica Campochiaro, Riccardo Casatta, Paolo Cremonesi, and Roberto Turrin. Do Metrics Make Recommender Algorithms? *2009 International Conference on Advanced Information Networking and Applications Workshops*, 0(March):648–653, 2009.

[14] Elisa Campochiaro, Paolo Cremonesi, and Roberto Turrin. Analysis of Recommender Systems Based on Implicit Datasets. *ACM Transactions on Knowledge Discovery from Data*, 5, 2008.

[15] Pedro Cano, Markus Koppenberger, and Nicolas Wack. *An industrial-strength content-based music recommendation system.* ACM Press, New York, New York, USA, August 2005.

[16] Iván Cantador, Alejandro Bellogín, and David Vallet. *Content-based recommendation in social tagging systems.* RecSys '10. ACM Press, New York, New York, USA, 2010.

[17] Paolo Colombo and Michele Giussani. Caratterizzazione degli utenti televisivi per la selezione di pubblicità mirata, 2011.

[18] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[19] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. *Performance of recommender algorithms on top-n recommendation tasks.* ACM Press, New York, New York, USA, September 2010.

[20] Paolo Cremonesi, Roberto Turrin, Eugenio Lentini, and Matteo Matteucci. An Evaluation Methodology for Collaborative Recommender Systems. *2008 International Conference on Automated Solutions for Cross Media Content and MultiChannel Distribution*, (March):224–231, 2008.

[21] Maurice De Kunder. http://www.worldwidewebsize.com/.

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

[23] Mukund Deshpande and George Karypis. Item-based top- N recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, January 2004.

[24] DVB Project. http://www.dvb.org/.

[25] M.A. Ghazanfar and A. Prugel-Bennett. A Scalable, Accurate Hybrid Recommender System. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 94–98. IEEE, January 2010.

[26] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[27] Google Inc. http://adwords.google.com/.

[28] GroupLens Research. http://www.grouplens.org/.

[29] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. *Explaining collaborative filtering recommendations*. ACM Press, New York, New York, USA, December 2000.

[30] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[31] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. IEEE, December 2008.

[32] IMDb.com Inc. http://www.imdb.com/.

[33] George Karypis. Evaluation of Item-Based Top- N Recommendation Algorithms. In *Proceedings of the tenth international conference on Information and knowledge management - CIKM'01*, page 247, New York, New York, USA, October 2001. ACM Press.

[34] Yehuda Koren. *Factorization meets the neighborhood*. ACM Press, New York, New York, USA, August 2008.

[35] Yehuda Koren. The BellKor Solution to the Netflix Grand Prize. *Baseline*, (August):1–10, 2009.

[36] S K Lam, Dan Frankowski, and John Riedl. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. *Lecture Notes in Computer Science*, 3995:14–29, 2006.

[37] Last.fm Ltd. http://www.last.fm/.

[38] G Linden, B Smith, and J York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[39] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based Recommender Systems: State of the Art and Trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[40] H Mak, I Koprinska, and J Poon. Intimate: a web-based movie recommender using text categorization. In *Proceedings of the 2003 IEEEWIC International Conference on Web Intelligence*, number Imd, page 602. Citeseer, 2003.

[41] Matteo Matteucci. Lecture Notes on Natural Computation. 2008.

[42] P Melville, R J Mooney, and R Nagarajan. Content-boosted collaborative filtering for improved recommendations. *Proceedings Of The National Conference On Artificial Intelligence*, (July):187–192, 2002.

[43] Cataldo Musto. *Enhanced vector space models for content-based recommender systems*. ACM Press, New York, New York, USA, September 2010.

[44] Netflix Inc. http://www.netflixprize.com/.

[45] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems*, 54(1-7):1–17, 1998.

[46] M Papagelis and D Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence*, 18(7):781–789, 2005.

[47] M J Pazzani, J Muramatsu, and D Billsus. Syskill & Webert: Identifying interesting web sites. In Dan Weld and Bill Clancey, editors, *AAAIIAAI Vol 1*, volume 1, pages 54–61. AAAI Press / MIT Press, 1996.

[48] Michael Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin / Heidelberg, 2007.

[49] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5-6):393–408, December 1999.

[50] M F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[51] N Ramakrishnan, B J Keller, B J Mirza, A Y Grama, and G Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–63, 2001.

[52] P Resnick, N Iacovou, M Suchak, P Bergstrom, and J Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In R Furuta and C M Neuwirth, editors, *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, volume pp, pages 175–186. ACM, ACM, 1994.

[53] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, March 1997.

[54] James Salter and Nick Antonopoulos. CinemaScreen recommender agent: combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, 2006.

[55] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Series in Computer Science. Addison-Wesley, 1989.

[56] Gerard Salton, Edward A. Fox, and Harry Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[57] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. *Item-based collaborative filtering recommendation algorithms*. ACM Press, New York, New York, USA, April 2001.

[58] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce - EC '99*, pages 158–166, New York, New York, USA, November 1999. ACM Press.

[59] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 02*, (Sigir):253, 2002.

[60] Upendra Shardanand and Pattie Maes. *Social information filtering*. ACM Press, New York, New York, USA, May 1995.

[61] Barry Smyth and Paul Cotter. A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.

[62] Stephan Spiegel, Jérôme Kunegis, and Fang Li. Hydra. In *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management - CNIKM '09*, page 75, New York, New York, USA, November 2009. ACM Press.

[63] Panagiotis Symeonidis. Content-based Dimensionality Reduction for Recommender Systems.

[64] Elaine G Toms. Serendipitous Information Retrieval. *Library and Information Science*, (1968):11–12, 1999.

[65] B Towle and C Quinn. Knowledge Based Recommender Systems Using Explicit User Models. *Knowledge Creation Diffusion Utilization*, pages 00–04, 2000.

[66] S M Weiss, C A Kulikowski, S Amarel, and A Safir. A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, 11(1-2):145–172, 1978.

[67] Yahoo! Inc. http://www.delicious.com/.

[68] Yahoo! Research. http://sandbox.yahoo.com/.

[69] Bo Yang, Tao Mei, Xian-Sheng Hua, Linjun Yang, Shi-Qiang Yang, and Mingjing Li. Online video recommendation based on multimodal fusion and relevance feedback. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 73–80. ACM, 2007.