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

Integrazione delle informazioni sociali nei sistemi di raccomandazione

Relatore: Prof. Paolo Cremonesi
Correlatore: Dott.ssa Elena Baralis
Correlatore: Dott. Roberto Turrin

Tesi di Laurea di:
Roberto Pagano, matricola 768803
Michele Russo, matricola 755648

Anno Accademico 2011-2012

Sommario

I *sistemi di raccomandazione* sono degli strumenti informatici in grado di predire gusti e preferenze di un utente al fine di selezionare un insieme personalizzato di prodotti o servizi. L'informazione relativa ai legami di amicizia, tipicamente disponibile nei social network, può migliorare la qualità della raccomandazione, dato che il consiglio di un amico ricopre un ruolo fondamentale nei processi decisionali.

Lo scopo della tesi è quello di cercare di integrare le informazioni relative a legami di amicizia, o *informazioni sociali* nel processo di raccomandazione. Sono state valutate due diverse metodologie di integrazione, validate su tre dataset pubblici: *social modeling* e *social preprocessing*. La prima tecnica *apprende* le informazioni sociali direttamente nel *modello* dell'algoritmo di raccomandazione, mentre la seconda elabora i dati di ingresso al sistema sulla base delle informazioni sociali, senza modificare la fase di apprendimento del modello.

L'integrazione tramite *social modeling* non ha prodotto risultati significativi in nessuno dei tre dataset utilizzati. Il *social preprocessing* ha prodotto invece miglioramenti in termini di recall, soprattutto sul dataset di *Delicious* in cui la qualità degli algoritmi dello stato dell'arte è stata notevolmente migliorata: l'algoritmo *PureSVD*, ad esempio, ha ottenuto un miglioramento in termini di recall di 0.47 (calcolata in una lista di 10 item raccomandati).

Indice

Capitolo 1

Introduzione

I *sistemi di raccomandazione* sono degli strumenti informatici in grado di predire le preferenze e i gusti di un utente. Infatti l'evoluzione dei sistemi informativi e la diffusione capillare del Web hanno messo a disposizione degli utenti una grande quantità di informazioni, quantità così imponente da risultare una sfida in termini di *sovraccarico cognitivo*. In altre parole la capacità di un utente di prendere una decisione può essere compromessa dalla presenza di troppe informazioni [89]. Per questo motivo è sorta la necessità di sistemi software in grado di assistere gli utenti nella navigazione di imponenti quantità di dati.

La raccomandazione si applica in molteplici processi decisionali, come quali oggetti comprare, quale musica ascoltare o quali notizie leggere. *Item* è il termine generale usato per denotare cosa il sistema raccomandi agli utenti (es. film, libri, vacanze). Un *sistema di raccomandazione* normalmente è pensato per un tipo specifico di item, per esempio CD o notizie, ed il suo design, la sua interfaccia grafica e la tecnica di raccomandazione sono specificatamente tarate per il tipo particolare di item oggetto della raccomandazione, in modo tale da fornire suggerimenti utili ed efficaci[68]. I sistemi di raccomandazione sono diretti verso individui che non hanno l'esperienza o la competenza per valutare un numero potenzialmente elevato di item che un sito web, ad esempio, può offrire[75].

Alla base di ogni sistema di raccomandazione vi è un set di informazioni da cui è possibile desumere le preferenze degli utenti verso determinati oggetti. Queste preferenze sono solitamente identificate come *rating*. Tali *rating* possono essere esplicitamente espressi (ad esempio il giudizio espresso da un utente per un per un brano musicale in una scala da uno a cinque), o dedotti implicitamente dalle interazioni di un utente con uno specifico oggetto (ad esempio l'ascolto di un brano). L'informazione implicita, ovviamente, è meno accurata di quella esplicita, ma è quella più facilmente reperibile nell'assenza o impossibilità ad avere informazioni esplicite. Esse possono essere inferite tramite l'analisi del

clickstream o dei pattern di navigazione dell'utente o dalla semplice interazione, come la semplice visualizzazione di un video o della descrizione di un hotel.

La categoria più popolare e diffusa di sistemi di raccomandazione si basa sul principio secondo il quale gli utenti tendono a preferire item che interessano ad utenti a loro simili (*Collaborative Filtering*); tale concetto di similarità è calcolato in base ai rating che gli utenti hanno in comune (siano questi impliciti o espliciti). La recente diffusione dei social network modifica questa prospettiva, consentendo agli utenti di stabilire esplicitamente legami di amicizia con altri utenti.

L'informazione relativa ai legami di amicizia può migliorare la qualità della raccomandazione, dato che il consiglio di un amico è tenuto in notevole considerazione nel contesto di una scelta. Questo recentemente ha portato alla nascita di una nuova famiglia di sistemi di raccomandazione, noti come *sistemi di raccomandazione sociali* (*Social Recommender System*); in questi sistemi le raccomandazioni sono basate sulle preferenze degli utenti con cui l'utente target ha un legame (sia questo di amicizia, fiducia o altro). Proprio questo è il principio su cui si basa questa ricerca.

Nel nostro studio ci siamo serviti di tre dataset pubblici reperibili online. I dati a disposizione, estratti dai social network *LastFM*, *Delicious* ed *Epinions* contengono sia informazioni sui rating, sia sui legami di amicizia tra gli utenti.

Le informazioni sociali possono essere integrate nel sistema di raccomandazione seguendo tre diversi approcci, come schematizzato in Figura 1.1: (i) modificando i dati in ingresso al sistema (*social preprocessing*), (ii) inserendole nella fase di apprendimento degli algoritmi di raccomandazione (*social modeling*) e (iii) combinando i risultati della raccomandazione con le relazioni sociali (*social postprocessing*).

Il contributo del nostro lavoro è duplice e riguarda l'integrazione di informazioni sociali tramite due dei tre approcci descritti: *social modeling* e *social preprocessing*.

Nell'integrazione tramite *social modeling* abbiamo preso in considerazione CLiMF[79] e ALSPR[81] - due noti algoritmi di riferimento nel dominio dei rating impliciti binari - estendendoli al fine di integrare informazioni sociali; si è proceduto, inoltre, alla modifica dell'algoritmo LTR[50] (che già nella versione originale fa uso di informazioni sociali) apprendendo, per ogni utente, in che misura debbano contribuire la raccomandazione di tipo collaborativo e la componente sociale. Le modifiche agli algoritmi apportate in questa fase del lavoro si sono rivelate più efficaci delle versioni originali, tuttavia il miglioramente non è sostanziale: ad esempio *CLiMF* ha un incremento in termini di recall di 0.04 nel caso migliore, mentre per *ALSPR* i miglioramenti sono pressocché irrilevanti.

Nella seconda parte del lavoro abbiamo studiato una soluzione alternativa denominata *social preprocessing*, e basata sull'elaborazione dei rating in ingresso al

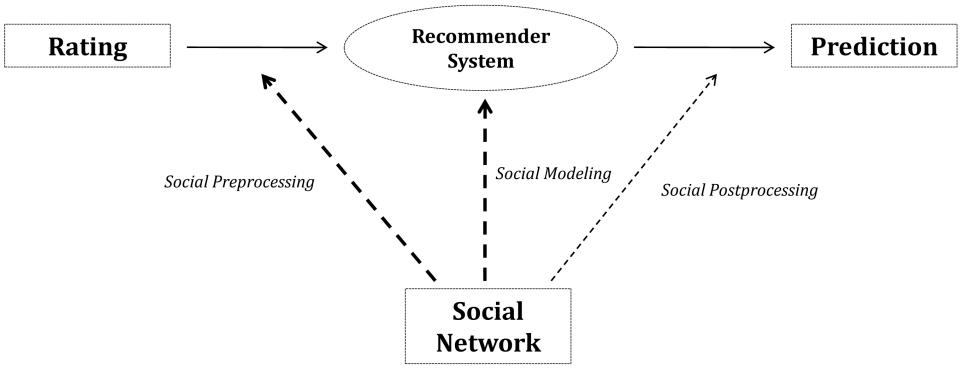


Figura 1.1: Integrazione delle informazioni sociali nel flusso di raccomandazione. Nella tesi sono state approfondite le tecniche di Social Preprocessing e Social Modeling.

sistema di raccomandazione; tale tecnica è utilizzabile con tutti i classici algoritmi di raccomandazione collaborativi. Con tale metodologia la matrice dei rating degli utenti viene resa più densa, il che contribuisce ad attenuare le difficoltà di apprendimento dei classici algoritmi di raccomandazione collaborativa in presenza di matrici di rating sparse. Il *social preprocessing* si è rivelato in generale vincente sugli algoritmi di partenza in termini di *recall*, mantenendo un livello di *fallout* quasi inalterato nei tre diversi dataset: ad esempio il *social preprocessing* applicato all'algoritmo *PureSVD* nel dataset di *delicious* ha dato un miglioramento in termini di recall di 0.43 rispetto alla versione originale, peggiorando il fallout di soli 0.05.

Struttura della tesi. Nel Capitolo 2 verrà brevemente introdotta la storia dei sistemi di raccomandazione con particolare attenzione agli algoritmi sociali ed impliciti. Nel Capitolo 3 verranno descritti gli algoritmi utilizzati per il confronto con lo stato dell'arte. Nel Capitolo 4 verranno descritti i dataset utilizzati e nel Capitolo 5 la metodologia utilizzata per confrontare gli algoritmi. Nel Capitolo 6 verrà esaminato l'impatto del fattore sociale nel modello. Nel Capitolo 7 verrà desrcitta l'operazione di preprocessing della *user rating matrix* per integrare le informazioni sociali all'interno degli algoritmi esaminati. Si conclude infine con alcune considerazioni sui risulati e possibili evoluzioni del nostro lavoro nel Capitolo 8. Per maggiore chiarezza nell'Appendice A è riportata una descrizione dettagliata della terminologia e della notazione utilizzata. Nell'Appendice B sono riportati per completezza tutti i grafici che, per brevità di trattazione, non sono stati inseriti all'interno del documento.

Capitolo 2

Stato dell'arte

Di seguito verrà presentata una breve descrizione della storia dei sistemi di raccomandazione. Successivamente si andrà nel dettaglio descrivendo cosa sono i sistemi di raccomandazione basati su informazioni sociali e i relativi lavori. Nella terza sezione verranno introdotti i sistemi di raccomandazione basati su rating impliciti. Infine verrà mostrata una tabella comparativa che riassume gli algoritmi esaminati.

2.1 Storia dei sistemi di raccomandazione

I sistemi di raccomandazione nascono all'inizio degli anni '90 come strumento per il filtraggio dell'informazione (*information filtering*), di particolare utilità in domini in cui la quantità di oggetti tra cui scegliere (es., i libri di una biblioteca) impedisce agli utenti di trovare quelle rilevanti (es., un interessante libro da leggere).

I sistemi di raccomandazione giocano un ruolo importante in siti importanti e di grosse dimensioni come Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, e IMDb. Netflix, un servizio di noleggio di film online, ad esempio, ha premiato con un milione di dollari il primo team che ha migliorato con successo le performance del suo sistema di raccomandazione[55].

Vi sono conferenze dedicate e workshop nel campo dei sistemi di raccomandazione, come l'ACM Recommender Systems (RecSys), fondata nel 2007 e ora divenuta il principale evento annuale per la ricerca sui sistemi di raccomandazione e sulle loro applicazioni. Inoltre, sessioni dedicate ai sistemi di raccomandazione sono frequentemente incluse in conferenze più tradizionali sull'argomento delle basi di dati, sistemi di informazione e sistemi adattivi. Tra queste vale la pena citare l'ACM Special Interest Group on Information Retrieval (SIGIR), User

Modeling, Adaptation and Personalization (UMAP), and ACM's Special Interest Group on Management Of Data (SIGMOD).

Vi sono stati riferimenti importanti in giornali accademici riguardanti la ricerca e lo sviluppo in quest'area. Tra i giornali che si sono occupati di più dell'argomento vi sono AI Communications (2008), IEEE Intelligent Systems (2007), International Journal of Electronic Commerce (2006), International Journal of Computer Science and Applications (2006), ACM Transactions on Computer-Human Interaction (2005) e ACM Transactions on Information Systems (2004).

Col Web 2.0 le attività online non sono più confinate alla semplice ricerca e alla navigazione. Il web si è evoluto così da permettere agli utenti di creare e condividere rapidamente dei contenuti, quindi contenuti non più in sola lettura. Gli utenti partecipano attivamente ai social network, caricano foto personali, scrivono blog, commentano e annotano informazioni prodotte da altri[60]. Questo concetto introduce svariate applicazioni nel mondo reale: ricerca e individuazione di comunità web, l'identificazione di argomenti salienti in una specifica comunità, raccomandazioni tempestive e pertinenti nell'ambito di applicazioni commerciali.

In questo contesto si sono diffuse principalmente due tecniche di raccomandazione: content-based-filtering [62] (CBF) e collaborative filtering [21](CF).

2.1.1 Content Based Filtering

L'approccio basato su contenuti suggerisce oggetti simili agli oggetti già apprezzati dall'utente. La rappresentazione delle caratteristiche può essere creata automaticamente se si tratta di item processabili automaticamente (news, articoli...), altrimenti è necessario inserirle manualmente (musica, video, ...). Per molte tipologie di item, d'altro canto, è praticamente impossibile stabilire quale sia il set appropriato di caratteristiche in grado di rappresentare una descrizione oggettiva.

Le prime tecniche *content-based* derivano dalla ricerca su *information retrieval* [4] [71] ed and *information filtering* [7]. In seguito a recenti sviluppi in questi due ambiti, e alla crescente importanza di applicazioni basate su informazioni testuali, si è sviluppato un filone di sistemi di raccomandazione *content-based* orientato ad item contenenti informazioni testuali, per esempio url, news e messaggi. Il contenuto di questi sistemi è generalmente descritto con parole chiave. Alcuni metodi calcolano il profilo dell'utente come una media dei vettori dei contenuti [5] [45], come ad esempio fa l'algoritmo *Rocchio* [70]. In alternativa è possibile utilizzare un *Bayesian classifier* [64] per stimare la probabilità che un documento sia apprezzato. L'uso dell'algoritmo di *Winnow* [48] ha prodotto ottimi risultati in questo ambito, specialmente in situazioni in cui ci sono diversi tipi di caratte-

ristiche degli item da tenere in considerazione [63]. Sono state impiegate anche altre tecniche di *machine learning* quali il *clustering*, gli *alberi decisionali* e le *reti neurali* [64].

Le informazioni dei profili utente possono essere esplicitamente immesse dagli stessi utenti o implicitamente dedotte da agenti software che monitorano le attività degli utenti [23]. Attualmente l'informazione dei profili utente per la raccomandazione online è ottenuta principalmente dall'analisi dei log di utilizzo (clickstream, pattern di navigazione, ...). Tuttavia le inferenze desunte dalle interazioni degli utenti sono non sono sempre attendibili [40]. L'acquisizione esplicita è più accurata ma può gravare esageratamente sull'utente[57]. L'user profiling per sistemi di raccomandazione si fonda principalmente sui dati estratti dai rating degli utenti, tuttavia la densità dei rating disponibili in sistemi commerciali è spesso inferiore all'1% [73]: questo tipo di problema viene spesso indicato come *sparsità dei dati* o *cold start problem*[74]. Inoltre nell'ambito dell'e-commerce è sempre più diffusa la falsificazione dei dati; un attacco molto comune per questo tipo di sistemi consiste nel creare un gruppo di utenti fittizzi che assegnano pseudo ratings per avvantaggiare o sfavorire certi prodotti (*malicious rating*).

I sistemi di raccomandazione *content-based* sono piuttosto limitati in determinate circostanze [5] [76]. In prima istanza, queste tecniche sono limitate dalle caratteristiche esplicitamente associate agli *item* che si intende raccomandare. Sebbene sia presente una florida letteratura riguardo metodi per estrarre automaticamente dati di tipo testuale, altri domini sono meno adatti per questo tipo di analisi (si pensi a qualsiasi tipo di dati multimediali, immagini, video, audio, ...), quindi spesso è richiesto un contributo manuale per l'individuazione delle caratteristiche e da un punto di vista pratico questa è una grossa limitazione [76]. Si consideri inoltre che è impossibile distinguere due item diversi se i rispettivi insiemi di caratteristiche sono uguali. Questo problema è piuttosto evidente in domini di tipo testuale, infatti un sistema di raccomandazione non è in grado di distinguere un articolo ben scritto da uno di bassa qualità, poiché la sua analisi è strettamente limitata alle occorrenze dei termini nel testo ed ogni articolo è individuato esclusivamente da un set di parole chiave, quindi da informazioni poco rappresentative per poter inferire sulla qualità [76]. Un'altra questione importante è nota in letteratura come *over-specialization*. Si tratta di quel fenomeno per cui l'algoritmo tenderà a raccomandare *item* via via più simili a quelli già predetti. Si pensi ad esempio alla raccomandazione di film: un utente appassionato di film di azione difficilmente riceverà un suggerimento per un film romantico. Una soluzione che generalmente viene adottata è quella di introdurre delle componenti casuali nella raccomandazione, ad esempio adoperando algoritmi genetici [78]. In generale la varietà di raccomandazioni è una caratteristica desiderabile. Alcuni algoritmi, come DailyLearner[13], cercano di garantire una

certa varietà non solo tenendo in considerazione item simili ai gusti dell'utente, ma anche escludendo item troppo simili ad altri precedentemente presi in esame dall'utente. Nell'ambito della raccomanzione di documenti sono state proposte cinque misure di ridondanza per stabilire se un documento, oltre adessere rilevante, contenga anche nuove informazioni [92]. Un'altra grande limitazione dei sistemi *content-based* riguarda il numero di rating necessari affinché l'algoritmo sia in grado di effettuare una predizione. Un nuovo utente, quindi, avrà pochi rating, e di conseguenza la raccomandazione non potrà essere accurata. In letteratura ci si riferisce a questo problema come *new user problem.*

2.1.2 Collaborative Filtering

I sistemi di raccomandazione collaborativi (o *collaborative filtering systems*) suggeriscono item ad un utente basandosi su item a cui altri utenti hanno assegnato una valutazione. Si pensi ad uno scenario di raccomandazione in cui gli item sono brani musicali: un sistema di raccomandazione collaborativo cercherà di individuare gli utenti simili all'utente per cui intende effettuare la raccomandazione, quindi utenti con gusti musicali simili, e raccomanderà quindi i brani più apprezzati da questo set di utenti. Il *collaborative filtering* si basa sui rating degli oggetti valutati dagli utenti con un profilo simile a quello dell'utente e si fonda, quindi, sulla disponibilità di profili utente che contengono la storia pregressa dei rating degli utenti, in questo modo non è richiesta alcuna conoscenza sugli item e quindi nessun intervento umano. Per questo, ed altri motivi, è utilizzato con successo in svariate applicazioni (Amazon.com, Movielens, ...) [1].

Sono stati sviluppati parecchi sistemi di raccomandazione collaborativi. Uno dei primi sistemi è stato probabilmente Grundy [69], utilizzato per la raccomandazione di libri, il quale proponeva l'utilizzo di stereotipi per costruire modelli di utenti basandosi su una quantità limitata di informazioni. I primi sistemi ad utilizzare questo tipo di algoritmi per la predizione automatica furono GroupLens [42] [67], Video Recommender [29], e Ringo [76]. Altri noti esempi di *collaborative filtering* sono il sitema di Amazon per la raccomandazione di libri, PHOAKS per suggerire informazioni rilevanti sul web [83].

Algoritmi di tipo collaborativo possono essere raggruppati in due classi: *memory-based* e *model-based* [2]. Gli algoritmi *memory-based* [76] [12] [18] [59] [67] sono essenzialmente euristiche basate sull'insieme di preferenze precedentemente espresse dagli utenti, mentre gli algoritmi *model-based* [9] [12] [24] [25] [30] [52] [62] [85] utilizzano le preferenze espresse dagli utenti per apprendere un modello con cui poi effettuare le predizioni.

Questo tipo di algoritmi non soffrono della maggior parte dei problemi elencati per gli algoritmi *content-based*; essi, ad esempio, non dipendono dalla tipologia

degli item, e sono insensibili anche al caso in cui nuovi item siano diversi da item osservati in precedenza. D'altra parte anche i sistemi collaborativi soffrono di alcune debolezze. Analogamente a quanto detto per i sistemi *content-based*, è presente il problema del nuovo utente. Anche in quest'ambito sono state proposte numerose tecniche, gli approcci più promettenti sono ibridi, cioè approcci che combinano le caratteristiche delle due tipologie di algoritmi [65]. I sistemi collaborativi sono in difficoltà anche in presenza di nuovi item. Questi si basano esclusivamente sulle preferenze degli utenti, quindi è necessario che un item sia stato valutato da un certo numero di utenti, affinchè la raccomandazione sia efficace. Anche in questo caso gli approcci ibridi si sono dimostrati promettenti. La *sparsità* dei dati è una questione che in generale riguarda tutti i sistemi di raccomandazione, il numero di rating ottenuti è generalmente molto piccolo rispetto a quello che si vorrebbe predire. Si immagini uno scenario di raccomandazione di film: i film rivolti ad un pubblico di nicchia sono valutati solo da un'esigua porzione di utenti; per questo motivo il sistema li raccomanderà piuttosto raramente, anche nel caso in cui questi pochi utenti abbiano assegnato valutazioni molto alte. Si può facilmente pensare ad uno scenario duale in cui un utente abbia dei gusti particolari con quasi nessun punto in comune agli altri utenti, anche in questo caso la predizione sarà poco efficace [5]. Una soluzione che generalmente viene adottata per risolvere il problema della *sparsità* è quella di calcolare la similarità tra utenti utilizzando altri dati oltre quelli delle valutazioni, ad esempio i dati demografici: sesso, età, posizione geografica, istruzione e posizione sociale. Questa tecnica è nota come *demographic filtering* [63]. Un altro approccio importante per ovviare il problema della *sparsità* consiste nell'utilizzare una tecnica di riduzione della dimensionalità della matrice dei rating: *Singular Value Decomposition* (SVD) [9] [72].

2.2 Social Recommender Systems

Utilizzare informazioni sociali per suggerire raccomandazioni è un aspetto cruciale per capire i processi decisionali che spingono un utente a propendere per un oggetto piuttosto che per un altro. E' stato dimostrato che i prodotti suggeriti dagli amici sono preferiti a quelli suggeriti da un attendibile sistema raccomandazione [80]. Uno studio recente ha dimostrato che un passaparola positivo tra clienti è un eccellente fattore di predizione per la crescita di un'azienda. [76]

A differenza dei dati sui rating, che sono dati numerici, i contenuti generati dall'utente comprendono una varietà di forme di media (testi, audio, immagini, video, ...). Grazie al notevole contenuto semantico, offrono un grande potenziale per approfondire la conoscenza degli utenti, degli item e delle varie relazioni che intercorrono tra utenti e item. Le informazioni estratte possono mitigare il *cold*

start problem e i *malicious rating*. L'estrazione e l'analisi delle opinioni, come il ricapitolare le opinioni degli utenti [95], e l'analisi dei pareri delle recensioni prodotte dagli utenti [20], sono ottimi strumenti per migliorare la qualità delle raccomandazioni [82]. Alcune compagnie hanno adottato con successo questo tipo tecniche, tra queste Epinions e Amazon[94].

2.2.1 Breve storia dei social network

Per social network si intende un sito web dove si interagisce con utenti con cui si condividono interessi personali o professionali, o altre informazioni anagrafiche tra le quali il luogo di nascita o l'istituto in cui si ha studiato [21], o in altre parole un sito web che aiuta gli utenti ad incontrarsi, trovare punti in comune, comunicare e condividere contenuti, e costruire comunità . Al giorno d'oggi vi è una grande varietà di attività sociali sui siti internet: blogging, acquistare su siti commerciali, appuntamenti online, pubblicare informazioni personali[94].

I social network hanno origine nel desiderio di alcune persone di volersi riconnettere con vecchi amici di scuola. *Classmates.com* (1995) è considerato il primo social network, ha totalizzato 40 milioni di utenti nel giugno 2007. Un paio di anni dopo viene lanciato *SixDegrees.com*, che nonostante il successo non è riuscito ad affermarsi come un business sostenibile. Nel 1999 *Epinions.com* sviluppa il primo *trust-based* social network, i membri possono decidere se fidarsi o meno (*trust* o *distrust*) di un altro membro. Concetto che in seguito *Twitter* rivisiterà col concetto di *follower*. Nel 2002 nasce il primo social network di successo orientato al business: *friendster*. Sulla scia del suo successo seguono l'anno successivo *LinkedIn* e *Xing*, entrambi orientati al mondo del lavoro. Attualmente i siti più popolari in assoluto sono *Myspace* (2003), *Facebook* (2004), *Bebo* (2005), *Twitter* (2006) [21] e il recente *Google+* (2011).

Dato il successo dei sistemi di raccomandazione di item in siti commerciali (come Amazon e Netflix), si è pensato di rivedere il problema della raccomandazione nell'ottica dei social network. In generale, i sistemi di raccomandazione cercano di fornire suggerimenti mirati basandosi sul comportamento degli utenti e sulle informazioni che loro stessi hanno esplicitamente espresso nei propri profili. Tuttavia non è stata posta la giusta enfasi nella possibilità di utilizzare informazioni sociali per prevedere i gusti degli utenti [94]. Intuitivamente, quando un utente è interessato all'acquisto di un prodotto che non gli è familiare, generalmente tenderà a consultare un amico, o un conoscente che ha già utilizzato il prodotto, ottenendo così un consiglio immediato. Analogamente, quando si riceve un consiglio da un amico, si tende ad accettare la raccomandazione poichè è considerato una fonte attendibile[27]. Tuttavia è stato fatto presente che la relazione di amicizia in certi casi può essere fuorviante, perchè di fatto due utenti

hanno diverse ragioni per stringere una relazione di amicizia [90]. Si pensi ad un sito come *last.fm*, in cui gli utenti condividono i propri interessi musicali e si consideri un utente appasionato di *jazz* e di *musica classica*: possibilmente tra i suoi amici ci saranno sia appassionati di jazz sia di musica classica; in questo caso la relazione di amicizia può essere fuorviante perchè non è detto che gli amici appassionati di jazz siano anche interessati alla musica classica e questo naturalmente ha un impatto sulla qualità delle raccomandazioni. Sebbene l'impatto di questo scenario sia piuttosto esiguo, è un interessante spunto di ricerca; si potrebbero utilizzare, ad esempio, altre informazioni sociali come l'appartenenza ad un gruppo[90], oltre alla classica relazione di amicizia,

L'influenza sociale gioca un ruolo fondamentale nel momento della scelta di un prodotto. Molte strategie di marketing sfruttano questo aspetto della natura umana, ottenendo risultati sorprendentemente soddisfacenti. Un esempio particolarmente noto è quello di *Hotmail*, il servizio di posta elettronica: esso aggiunge un testo promozionale in coda ad ogni messaggio inviato, così da diffondere attraverso la rete di amicizie degli utenti il proprio servizio di posta elettronica. Questa manovra è risultata così efficace da far crescere gli utenti fino a 12 milioni in meno di 18 mesi[39].

I risultati presentati in [16] e in altri lavori simili confermano che i *social network* costituiscono una fonte di informazioni che può essere sfruttata per migliorare la qualità delle raccomandazioni.

2.2.2 Integrazione di informazioni sociali in algoritmi Collaborative Filtering

Il *collaborative filtering*, rispetto ad altre tecnologie di raccomandazione, ha la capacità di lavorare in domini dove gli attributi degli item sono difficili da ottenere o non possono essere estratti automaticamente. Inoltre può fornire raccomandazioni con *serendipità*, cioè che non sono simili agli item nel profilo utente ma che, sorprendentemente, lo interessano. Come già accennato prima, il *collaborative filtering* risente dei problemi di *cold start*, *scalabilità* e *sinonimia*. L'integrazione di informazioni sociali è stata largamente utilizzata per cercare di risolvere questi limiti[32]. Gli utenti di un sistema sociale non possono essere considerati come indipendenti e identicamente distribuiti, ma va tenuto conto della reciproca influenza sociale degli amici. Su questa assunzione sono stati sviluppati parecchi lavori dedicati alla integrazione di informazioni sociali in algoritmi di tipo collaborativo.

La raccomandazione di notizie ha recentemente ricevuto particolare attenzione, per via di un brevetto depositato da *Facebook* per la generazione dinamica di notizie[96]. Alcuni social network come *Facebook* e *MySpace* permettono agli

utenti di filtrare le azioni o gli utenti da cui provengono le notizie; questi possono quindi essere bloccati, così da non generare notizie. In ambienti di questo tipo le notizie vengono generalmente presentate cronologicamente dalla più alla meno recente, in questo modo però è molto probabile che si ottenga un'inondazione di notizie coprendo informazioni più interessanti. *Freyne* e *Berkovsky* propongono un lavoro [22] incentrato sulla raccomandazione di notizie rilevanti per utenti di un social network (*IBM's Social-Blue* [19]). A differenza di altri algoritmi che calcolano la rilevanza tra item con un metodo globale, questo cerca di individuare, per ogni utente, quali siano i criteri per cui degli item possano essere considerati rilevanti. Nello specifico vengono analizzate le interazioni sociali degli utenti, identificando dei modelli di interesse nel breve e nel lungo periodo, attraverso uno studio della cronologia. In questo modo è possibile delineare le azioni più incisive per ciascun utente e utilizzarle per stabilire quali siano gli item più rilevanti. Ogni *item* comprende quattro componenti: il *soggetto*, cioè chi compie l'azione, l'*azione*, che è causa un certo cambiamento nello stato della rete, l'*oggetto* sul quale l'*azione* è compiuta (ad esempio un utente per un'azione di amicizia, o un contenuto per un'azione di pubblicazione di contenuti), e il *tempo* in cui l'*azione* è stata compiuta. In questo lavoro viene tenuto conto solo dell'utente e dell'azione che riguardano l'*item* in esame. Viene proposta inoltre una metrica per quantificare il livello di interesse di un *item* per un utente. Da questo studio è stato possibile concludere che una buona strategia consiste nel combinare la rilevanza delle azioni desunta dai modelli a lungo e breve termine, piuttosto che propendere per uno dei due.

Jianming e Wesley [27] hanno proposto un paradigma per utilizzare le informazioni nei social network (in particolare preferenze degli utenti, gradimento degli item e influenza degli amici) per effettuare predizioni attraverso un modello probabilistico. L'analisi si basa su un dataset di grandi dimensioni estratto da un social netork online. Questa analisi ha rivelato che utenti amici tendono a selezionare gli stessi item e dare valutazioni simili[27]. I risultati sperimentali ottenuti da questo dataset mostrano che il sistema proposto non solo migliora l'accuratezza della predizione del sistema, ma risolve in parte il problema della *sparsità* e il *cold-start*. Inoltre viene proposto di migliorare le performance del sitema applicando un *semantic filtering* sui social network, ovvero preprocessare i dati a disposizione con un analizzatore semantico, in modo da arrichire il significato dei dati. Con questo esperimento viene dimostrato come è possibile estrarre amici rilevanti per fare inferenza sul significato e sul valore delle relazioni di amicizia. In termini di accuratezza della predizione, le informazioni addizionali ottenute sugli utenti e gli amici della rete sociale aiutano a comprendere il comportamento dei singoli utenti e l'impatto che questo ha sui rating. Inoltre è possibile modellare e interpretare gli utenti con più precisione e quindi ottenere

un miglioramento della precisione del sistema. Gli autori sostengono che, grazie alle informazioni di amicizia non è più necessario trovare utenti simili misurando le rispettive similarità di rating, poiché la relazione di amicizia indica che gli utenti hanno dei punti in comune. Quindi l'impatto della *sparsità* può essere per certi versi ridotto. Infine, per quanto riguarda il problema del *cold-start*, sebbene un utente non abbia un numero di rating sufficiente per effettuare una predizione è possibile integrare questa mancanza con le informazioni estratte dagli amici.

Matrix Factorization Tecniche di fattorizzazione delle matrici sono piuttosto utilizzate nell'ambito del *collaborative filtering*, e numerose sono le proposte presentate per migliorare questa tipologia di algoritmi con informazioni sociali[90][50][51][35][32]. Nella sezione precedente sono state illustrate le problematiche inerenti al fenomeno della *sparsità* nei sistemi di raccomandazione basati su *collaborative filtering*. Questo fenomeno colpisce prevalentemente i nuovi utenti, o gli utenti inattivi per un lungo periodo di tempo. Tuttavia la diffusione dei *social network* e la possibilità di essere integrati in siti di *e-commerce* si è rivelato essere uno strumento piuttosto potente per alleviare il problema della sparsità. Le relazioni sociali possono infatti, in certi casi, sopperire alla mancanza dei dati[90].

Ultimamente si è posta molta attenzione sui *trust-aware recommender systems* [53] [54]. La maggior parte di questi metodi è basata su alcune euristiche ad hoc e risentono ancora dei problemi di *sparsità* e *scalabilità*. Inoltre risultano ancora poco chiare le relazioni tra le matrici dei rating e le relazioni di *trust* tra gli utenti[50]. La relazione di trust può essere pensata come una relazione di amicizia asimmettrica e non necessariamente binaria. Basandosi sull'assunzione che la relazione di similiarità tra i profili degli utenti sia in correlazione con la relazione di amicizia, gli autori di *Learning to recommend with social trust ensemble*[50] propongono un modello più realistico per la raccomandazione. In particolare sono tre le assunzioni su cui è incentrato il lavoro: ogni utente ha le proprie caratteristiche, cioè gusti diversi per diversi item (film, libri, musica, cibo); in secondo luogo ogni utente può facilmente essere influenzato dagli amici in cui ripone fiducia, e quindi tende ad avere gusti simili a quelli degli amici; infine la decisione finale dell'utente è basata su un equilibrio tra i propri gusti e quelli degli amici. Si tratta di un framework probabilistico che fonde i gusti degli utenti con quelli delgi amici. Col termine *Social Trust Ensemble* gli autori indicano la formulazione delle restrizioni sociali nell'ambito dei sistemi di raccomandazione. Gli autori sostengono inoltre che, a seguito di uno studio sulla complessità computazionale dell'algoritmo, questo si presta in ambienti con dataset di grosse dimensioni, e che le performance sono superiori rispetto ad altri metodi in letteratura. Basandosi sulla stessa assunzione degli autori di *Learning to recommend with social trust ensemble* Yang e King[51] presentano un approccio per integrare la *rete sociale*

con la *matrice dei rating*. Tale metodo è basato sull'analisi probabilistica dei fattori latenti (*latent features*), ovvero la rete sociale e la matrice dei rating sono connesse attraverso uno spazio di fattori latenti. Quindi è possibile effettuare raccomandazioni sociali in seguito all'apprendimento di questi fattori. Sono stati condotti dei test su un dataset estratto da *Epinions*; questi hanno dimostrato che l'algoritmo è piuttosto promettente soprattutto nei casi in cui l'utente ha pochi rating, se non nessuno. Inoltre l'analisi della complessità ha mostrato che l'algoritmo è particolarmente performante anche su dataset di scala maggiore.

Un contributo importante consiste nell'incorporare il meccanismo di propagazione della fiducia di un utente verso un altro (*trust propagation*). Il fenomeno di *trust propagation* si è rivelato essere cruciale sia nell'ambito delle scienze sociali, sia nell'analisi dei social network e di conseguenza anche nella raccomandazione *trust based*.[35] Con *SocialMF*[35] viene esplorato un approccio *model-based* basato sul techniche di fattorizzazione di matrici, per la raccomandazione di item all'interno di un *social network*. Questo approccio verte sull'ipotesi che si disponga di una rete sociale e che quindi sia possibile effettuare raccomandazioni per un utente basandosi sui rating degli utenti che condividono con lui, direttamente o indirettamente, delle relazioni sociali. Viene introdotto quindi un nuovo modello (*SocialMF*) basato su tecniche di fattorizzazione delle matrici che include il concetto di *trust propagation*. In sostanza il vettore delle caratteristiche di ogni utente (*feature vector*) è dipendente dai vettori dei sui vicini diretti. Ricorsivamente il vettore delle caratteristiche di ognuno di questi vicini dipenderà a sua volta dai vettori dei propri vicini. Questo effetto è mostrato nella distribuzione condizionale, considerando il vettore delle caratteristiche di un utente come una *distribuzione normale* attorno alla media dei *feature vector* dei suoi vicini. Sono stati condotti esperimenti su dataset reali: un dataset di pubblico dominio del sito di social bookmarking *Epinions.com*, e un dataset estratto dagli autori attraverso un *crawler* del sito *Flixster.com*, un servizio di social networking in cui gli utenti possono assegnare rating ai film. Gli esperimenti hanno dimostrato che il modello proposto riduce sensibilmente l'errore della raccomandazione (*Root Mean Squared Error*), soprattutto in presenza di utenti *cold start*. Ovviamente se un utente *cold start* non è connesso ad una rete sociale non si ha alcuna informazione aggiuntiva per migliorare la qualità della raccomandazione per l'utente.

In *Recommendations in Taste Related Domains Collaborative Filtering vs. Social Filtering* [26] vengono messi a confronto il *collaborative filtering* e il *social filtering*: la differenza tra i due sta nel modo in cui si calcola la similarità tra utenti come: nel collaborative filtering essa è uguale al prodotto scalare tra i loro vettori di rating $S_{uv} = \frac{r_u \cdot r_v}{\|r_u\| * \|r_v\|}$; nel social filtering la similarità è uguale al valore di trust che l'utente selezionato ha espresso verso i propri amici: $S_{uv} = t_{uv}$. Sono stati testati tre tipi diversi di algoritmi:

- *media semplice*: $r_{ui} = \frac{1}{|N_u|} \sum_{v \in N_u} r_{vi}$"
- *media pesata*: $r_{ui} = \overline{r_u} + \frac{1}{\sum_{v \in S_u} S_{uv}} \sum_{v \in S_u} (r_{vi} - \overline{r_v}) S_{uv}$
- *media non pesata*: $r_{ui} = \overline{r_u} + \frac{1}{|N_u|} \sum_{v \in N_u} r_{vi} - \overline{r_v}$

Questi tre algoritmi hanno performance migliori col social filtering, migliorando sia il *Mean Absolute Error* che la *F-measure*, soprattutto con utenti *cold start* (cioè con utenti con un numero di rating insufficienti per poter effettuare una raccomandazione soddisfacente).

Nell'articolo presentato da Rong e Pearl [32] viene indirizzato il problema del cold start, dal punto di vista della sparsità e della creazione di un nuovo utente. Vengono proposti tre approcci:
Il primo è un metodo di raccomandazione basato sulle informazioni sulla personalità dell'utente.
Il secondo è basato sulla combinazione lineare delle informazioni della personalità e delle informazioni sui rating: attraverso un parametro α viene mediata la similarità basata sui rating e quella basata sulla personalità.
Il terzo approccio utilizza un meccanismo a cascata per utilizzare entrambe le risorse: viene creata una rating matrix più densa, inserendo una parte dei rating mancanti con i rating predetti su utenti con profilo simile. Vengono effettuati due tipi di esperimenti per sondare i due tipi diversi di situazioni cold start. Il dataset utilizzato è quello di *discover music*, dal quale vengono filtrati gli utenti con meno di 20 rating. Questo dataset contiene anche informazioni sulla personalità. Il dataset è stato allargato utilizzando *last.fm*[1] e creando una funzione di rating apposita[32]. I metodi proposti migliorano il metodo di baseline (cioè il *Collaborative Filtering*) in termini di *Mean Absolute Error (MAE)* e *Relative Operating Characteristic (ROC)*, soprattutto il metodo a cascata, sia in situazioni di sparsità che di nuovo utente. Sebbene questo algoritmo non sia esattamente sociale, rappresenta comunque uno spunto interessante per risolvere il problema del cold start.

Graph Oriented Il *collaborative filtering* soffre del problema della sparsità e in più non ha informazioni sulla confidenza della raccomandazione. La raccomandazione *puramente sociale*, cioè basata sulla *trust* utilizza i dati della rete di trust ma, a causa della sparsità, si è costretti a considerare i rating di amici indiretti, i quali sono poco affidabili, il che può portare ad un decremento della precisione. Con *Trustwalker*[36] viene proposto un approccio alla raccomandazione basato sui grafi (e in particolare sull'algoritmo di *random walk* su un grafo)

[1] www.last.fm

che cerca di trovare un buon compromesso fondendo le informazioni collaborative con informazioni derivanti dalla trust. L'algoritmo consta di due componenti fondamentali: la *random walk* sulla rete di *trust*, che restituisce sempre un rating, e la selezione probabilistica degli item, che fa sì che la probabilità di usare il rating di un item simile a quello cercato aumenti con l'aumentare della lunghezza della random walk. *TrustWalker*, quindi, preferisce utenti ad una distanza minore, migliorando la precisione. Una singola *random walk* che parte dall'utente u_0 su un item i può essere descritta come di seguito: ad un certo step k della random walk ci troveremo in un certo nodo u; se esiste r_{ui} esso viene utilizzato e la random walk termina. Se r_{ui} non esiste, abbiamo due possibilità: con probabilità $\phi_{u,i,k}$ viene selezionato r_{uj} con j simile ad i e la random walk termina; con probabilità $1 - \phi_{u,i,k}$ si continua la random walk su un utente che è un diretto vicino dell'utente u. Le similarità tra item vengono calcolate col *coefficiente di correlazione di Pearson* mentre la probabilità $\phi_{u,i,k}$ cresce all'aumentare di k e dipende dalla similarità massima tra l'item i e gli item dell'utente corrente allo step k. Per calcolare il rating $\hat{r}_{ui}$ vengono eseguite più random walk finchè la differenza tra la varianza dei risultati scende sotto un valore minimo ϵ. Trustwalker riesce a migliorare la copertura delle raccomandazioni trust-based, mantenendo la stessa precisione o addirittura migliorandola leggermente e supera il collaborative filtering in termini di copertura.

Un altro algoritmo basato sui grafi è presentato in *Social network-based recommendation: a graph random walk kernel approach* [47]. In questo articolo viene affrontato il problema della raccomandazione basandosi esclusivamente sulle informazioni sociali di un utente. Utilizzando un approccio *kernel-based* vengono catturate le similarità principali tra utenti in un *random walk kernel su grafi*, quindi viene costruito un un modello su *Support Vector Regression* per predire le opinioni degli utenti su prodotti. Si assuma di avere una rete sociale $G = (V, E)$, dove V è l'insieme degli utenti ed E l'insieme delle relazioni tra utenti, e un sottoinsieme di una matrice dei rating $R(V, P)$ sui prodotti P. Lo scopo di una raccomandazione basata su social network è di colmare la parte mancante della matrice dei rating. Sono due i principi sociali alla base di questo algoritmo: in primo luogo l'influenza reciproca tra gli utenti di una rete attraverso il passaparola, ricerche precedenti hanno dimostrato che lo sviluppo di modelli *trust based* riassume propriamente questo concetto [87]; in secondo luogo persone con caratteristiche simili tendono a diventare amiche, fenomeno definito come omofilia [56]. Sono stati condotti esperimenti su un dataset estratto da un sito di recensioni di film. Il modello proposto dagli autori risulta superiore ad altri modelli *trust based* e *kernel basati su grafi*. La metrica usata per calcolare la bontà degli algoritmi è il *Mean Absolute Error*, cioè la media della differenza assoluta tra rating reale e rating predetto.

2.2.3 Annotation Based

Il tagging collaborativo è una delle tecniche di categorizzazione più comuni del *Web 2.0*. Di seguito verranno presentati degli algoritmi che riguardano la raccomandazione di tag o di item nelle *folksonomie*. Una *folksonomia* descrive una categorizzazione di informazioni generata dagli utenti mediante l'utilizzo di parole chiave, o *tag*, scelte liberamente. Nel campo dei sistemi di raccomandazione essa può essere formalizzata con un tensore a tre dimensioni (utenti, item e tag) che estende la matrice dei rating a due dimensioni. *delicious*[2], *last.fm*[3] e *flickr*[4] possono essere considerati tre esempi di folksonomie.

Con la diffusione dei sistemi di raccomandazione, si è diffuso l'utilizzo dei *tag*. Un *tag* è una parola (aggettivo, sostantivo, verbo) che cerca di descrivere meglio l'oggetto della raccomandazione, oppure cerca di descrivere cosa quell'oggetto significa per l'utente.

É da notare come nelle folksonomie molto spesso sono presenti rating unitari, spesso *impliciti*, che indicano la presenza o meno di un tag per l'item di interesse da parte di un utente.

L'analisi dei tag può aiutare a generare raccomandazioni migliori; se però l'analisi dei tag non tiene conto della semantica, si incorre in problemi quali la *polisemia* e la *sinonimia*; senza considerare la semantica, inoltre, è impossibile differenziare i vari interessi sociali che gli utenti vogliono descrivere con lo stesso tag.

Nell'articolo[41] si introduce un nuovo modello che cerca di catturare la semantica di tag generati dall'utente e propone un sistema di raccomandazione sociale che incorporla. L'approccio proposto dagli autori cerca prima di determinare item semanticamente simili utilizzando tag orientati alla semantica e dopo scopre semanticamente gli item che con più probabilità soddisfano i bisogni dell'utente. I loro esperimenti mostrano che il modello semantico può fornire performance migliori nella raccomandazione, rispetto ad altri metodi della letteratura, come *user based* e *item based collaborative filtering* e *most popular tags*.

Nell'articolo [88] viene utilizzata la *Probabilistic Latent Semantic Analysis* (PLSA) invece del collaborative filtering. La *PLSA* è stata introdotta da Hofmann[31] ed è stato dimostrato che migliora la qualità della raccomandazione in vari campi, partendo dal presupposto che i dati siano originati da un modello sottostante a meno dimensioni. L'approccio utilizzato dagli autori estende la *PLSA* in modo che il modello sia stimato dagli item dell'utente e dai tag in parallelo.

[2] www.delicious.com
[3] www.last.fm
[4] www.flickr.com

I test vengono eseguiti su un dataset di delicious. I due approcci (collaborative filtering e annotation based) vengono fusi mediando attraverso un parametro α e vengono stimati i parametri tramite l'algoritmo di Expectation Maximization (EM). I risultati migliori comunque sono quelli con $\alpha = 1$, cioè quelli basati sulle annotazioni (*tag*).

L'articolo[38] confronta alcuni algoritmi: il collaborative filtering adattato alle folksonomie, il metodo dei tag più popolari e *FolkRank*, un adattamento del *PageRank* alle folksonomie. Tutti gli algoritmi superano in performance il metodo dei tag più popolari, ma il confronto è vinto da *FolkRank* che dà risultati uniformemente migliori in ogni dataset. Ciò è dovuto al fatto che il *collaborative filtering* non è pensato per lavorare con tag o con rating impliciti: la funzione di rating scelta, infatti, pone il rating a 1 se è presente almeno un tag, 0 altrimenti, generando in questo modo dei rating che si discostano dalla volontà dell'utente. L'utente infatti potrebbe assegnare ad un item il tag 'inutile', ma l'algoritmo valuta questo item come rilevante per l'utente.

Lo scopo di un *sito di bookmarking* è quello di permettere agli utenti di salvare, organizzare e ricercare *bookmark* di pagine web. Ogni utente può aggiungere delle annotazioni testuali usando *tag* informali (cioè non legate a un dizionario specifico) o *metadati* di altro genere, ad esempio titoli e descrizioni. Nell'articolo[11] vengono esplorati metodi per la raccomandazione in un ambiente di *bookmarking* con componenti sociali, cioè di trovare nuove vie per la predizione di *bookmark* non ancora visionati ad utenti basandosi sulle informazioni dei rispettivi profili.

Una strada consiste nell'utilizzare i *metadati* a disposizione per migliorare il processo di raccomandazione. L'utilizzo dei metadati è abbastanza comune in letteratura, si tratta però di metadati legati agli item, cioè di informazioni strettamente inerenti al contenuto degli item, per cui in linea di massima comuni a tutti gli utenti[84]. L'innovazione consiste nell'incorporare *tag* e *metadati* in un algoritmo basato *nearest-neighbour collaborative filtering* e di integrare le metriche per il calcolo della similarità, utilizzate solitamente in letteratura, con metriche di similarità tra tag[58]. Inoltre sono stati condotti esperimenti su algoritmi *content-based* utilizzando i *metadati* per effettuare la predizione di item, basandosi direttamente sulla divergenza di *Kullback-Leibler*[73]. Gli esperimenti sono stati condotti su dataset di dominio pubblico, con un set di metriche standard così da rendere più semplice la verifica dei dati [13]. I dataset esaminati provengono da *Delicious*, *CiteULike* e *BibSonomy*.

Di seguito verranno trattati alcuni algoritmi che utilizzano le annotazioni per generare raccomandazioni sociali

Gli autori di *Interactive Recommendations in Social Endorsement Networks*[46] tratta la raccomandazione nelle *social endorsement networks*, cioè reti sociali in

cui è possibile incorporare entità come video, foto o anche altri utenti. Viene formalizzato il problema della raccomandazione interattiva in queste reti: data una sequenza di tag, il problema è quello di raccomandare entità che corrispondano alla sequenza e abbiano anche un numero significativo di utenti che le incorporano (*endorsers*); su *facebook*[5] l'operazione di endorse corrisponde al like oppure alla pubblicazione di una entità, su *citeULike*[6] corrisponde all'aggiunta di un articolo alla libreria personale dell'utente, su *twitter*[7] corrisponde all'operazione di *follow*.

Nell'articolo[46] la *social endorsement network* è modellata con un grafo bipartito con archi che vanno dagli utenti alle entità. Per *gruppo*, si intende un insieme di utenti che hanno incluso entità comuni. L'algoritmo consta di due parti: la prima prende in input l'insieme totale di gruppi e manda in output l'insieme filtrato di gruppi non ridondanti; la seconda parte prende in input la sequenza di tag e manda in output i gruppi che hanno la corrispondenza più alta con la sequenza data. L'algoritmo è stato testato col dataset di *DBLP*[8] e *twitter* e riesce a raccomandare entità popolari che corrispondono alla sequenza di tag immessa.

Di seguito verranno presentati degli algoritmi che utilizzano sia la *fattorizzazione di matrici* che i *tag* per generare raccomandazioni

Negli ultimi anni, i sistemi di raccomandazione CF sono stati largamente sviluppati ed adottati per supportare efficacemente il processo decisionale dell'utente, specialmente quando questo è posto di fronte ad un'ampia gamma di opzioni. Con la diffusione dei social network, abbiamo due ulteriori relazioni che possono aiutare nel processo di raccomandazione: l'amicizia e l'appartenenza al gruppo. Come sostenuto in [44] le relazioni sociali esplicite possono sicuramente essere applicate per compensare la limitazione dell'approccio inferenziale delle relazioni implicite e migliorare la qualità della raccomandazione. Inoltre l'appartenenza ad un gruppo, secondo gli autori dell'articolo[91], contiene più informazioni riguardanti le preferenze di un utente, rispetto all'amicizia. Gli autori conducono degli esperimenti fondendo le informazioni sociali attraverso due metodologie: la prima è combinare le informazioni sull'amicizia e sulla membership in una matrice della similarità pesata; la seconda consiste nell'inserire queste informazioni sociali in un grafo e poi utilizzare la random walk. Gli esperimenti mostrano un miglioramento dell'8% nella accuratezza delle raccomandazioni. Inoltre gli autori sostengono che la struttura a grafo si presta meglio ad utilizzare le informazioni sociali.

[5] www.facebook.com
[6] www.citeulike.org
[7] www.twitter.com
[8] http://kdl.cs.umass.edu/data/dblp/dblp-info.html

La raccomandazione di *tag* riveste particolare importanza nel momento in cui l'utente sta per assegnare un tag ad un item: in base alla storia dei suoi tag passati e in base ai tag associati da altri utenti all'item, viene generata una lista di tag suggeriti, in modo da facilitare la categorizzazione e la ricerca.

L'articolo *Pairwise interaction tensor factorization for personalized tag recommendation*[66] tratta la decomposizione di tensori (le folksonomie) per la raccomandazione di tag ad un certo utente verso un certo item. Essa si basa sulla *decomposizione di Tucker*, apprendendo il modello attraverso la tecnica del *Bayesian Personalized Ranking*. Gli autori riescono a ridurre la complessità della decomposizione di Tucker da cubica a lineare. Il loro metodo modella esplicitamente le relazioni utente-tag-item a coppie ed è basato sui grafi. L'algoritmo riesce ad ottenere le stesse performances della decomposizione di Tucker, migliorando addirittura le performance in alcuni dataset.

2.2.4 Gruppi Sociali

L'analisi delle reti sociali è un fenomeno largamente trattato negli ultimi anni. In molti *social network*, oltre alle relazioni di amicizia tra utenti, è presente il fenomeno di aggregazione in gruppi, o comunità, per condividere interessi comuni. Queste due reti esistono simultaneamente: la rete di amicizie tra utenti, e la rete di affiliazione di utenti nei gruppi. La formazione di gruppi e la conseguente evoluzione nelle reti sociali [3], è stata recentemente oggetto di studio, assieme alla concorrente evoluzione di reti di affiliazione [93]. Uno degli argomenti più interessanti da affrontare nell'ambito dell'analisi dei *social network* è la raccomandazione di affiliazioni, cioè suggerire ad un utente di prendere parte a un gruppo (o a una comunità) che possa interessargli. Il fatto che le reti sociali e di affiliazioni evolvano parallelamente, suggerisce che una buona soluzione al problema della raccomandazione di affiliazioni possa essere ottenuta dalla rete di amicizie, considerata assieme alla rete di affiliazione. In generale l'affiliazione non deve essere necessariamente inerente ad una comunità; si può sfruttare lo stesso concetto per applicazioni diverse, per esempio, più in generale per prederire i gusti di un utente.

Anche nell'articolo già trattato [91] viene trattato il fenomeno dei gruppi sociali, ma in quel caso non vengono raccomandate affiliazioni a gruppi, ma si utilizzano le informazioni dei gruppi per migliorare la qualità delle raccomandazioni.

Gli autori dell'articolo [86] propongono un modo semplice per combinare le due reti, e suggeriscono due modelli di affinità tra comunità di utenti con lo scopo di effettuare raccomandazioni di affiliazioni: il primo si basa sulla prossimità all'interno di un grafo, mentre il secondo utilizza i fattori latenti per modella-

re utenti e comunità. Gli algoritmi sono valutati su due dataset reali: *Orkut* e *Youtube*. Viene proposta inoltre una modalità di valutazione delle raccomandazioni, misurando la bontà delle prime 50 raccomandazioni per un utente medio. L'algoritmo basato sulla prossimità dei grafi risulta essere il più promettente dei due.

Gli esperimenti mostrano come la relazione di amicizia sia di per sé un buono strumento per migliorare la precisione di raccomandazione, e che fondere i contributi delle due relazioni sia un ulteriore miglioramento rispetto alla raccomandazione collaborativa classica. Yuan e Zhao [90] propongono l'analisi di un nuovo tipo di relazione sociale: l'appartenenza a un gruppo (o *membership*). Viene mostrato come la *membership* sia un forte indice di contenuto informativo, e quali sono gli effetti che questa ha sulle raccomandazioni se combinata con la classica relazione di *amicizia*. La differenza sostanziale tra *membership* e *amicizia* consiste nel fatto che la prima coinvolge due tipi di entità: *utenti* e *gruppi*, mentre la relazione di *amicizia* comprende soltanto gli *utenti*. Inoltre l'appartenenza ad un gruppo è generalmente un indicatore di uno specifico interesse dell'utente verso l'argomento trattato dal gruppo, al contrario l'amicizia può essere vaga e controproducente perchè due utenti possono avere diverse ragioni per essere amici [6]. Queste due relazioni sociali sono fuse in un sistema di tipo collaborativo con una tecnica di fattorizzazione. A fronte delle proprietà delle due relazioni, sono state adottate due diverse strategie di integrazione: *regolarizzazione* per la relazione di amicizia e *fattorizzazione collettiva di matrici* per incorporare la relazione di *membership*. Infine viene proposto un modello unico per combinare le due relazioni e ne vengono testati i risultati in un dataset di grosse dimensioni a cinque livelli di sparsità

2.2.5 Elicitation

Gli autori dell'articolo[49] mostrano una soluzione particolarmente interessante al problema del nuovo utente. Abbiamo precedentemente mostrato alcune soluzioni al problema note in letteratura, si veda ad esempio Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites, che propone di sopperire alla mancanze di informazioni sui rating utilizzando dei metadati. Va tenuto presente che non sempre è possibile reperire questi metadati, o in generale non sempre sono disponibili informazioni aggiuntive. Un'alternativa che si è dimostrata essere particolarmente valida è quella dell'elicitazione dei rating per gli utenti *cold start*, ovvero se il sistema non ha a disposizione abbastanza informazioni per calcolare una predizione chiede all'utente di esprimere un giudizio su alcuni item specifici (*seed*), così da poter calibrare la predizione sul nuovo utente. In passato sono stati proposti alcuni

metodi euristici per selezionare i *seed item* [65][77]. Gli autori di questo articolo vogliono fornire un modello preciso sul come selezionare i rating più utili per la raccomandazione.

Lo scopo di questo lavoro è quello di stabilire un approccio rigoroso per identificare gli utenti e gli item rappresentativi utilizzando tecniche di fattorizzazione su una matrice che indica quanto i dati siano rappresentativi. Il metodo proposto si è rivelato essere superiore sia per quanto riguarda la *copertura* e la *diversità* dei dati, sia per la scelta dei rating più rappresentativi, che di fatto si sono rivelati più utili (di circa il 10%) rispetto ad altri metodi concorrenti. Nello specifico l'approccio consiste nell'utilizzare il concetto di *maximal-volume* per l'approssimazione di matrici e di generare quindi un modello basato sulla fattorizzazione della *matrice di rappresentatività* (*RBMF*).

2.3 Sistemi di raccomandazione con rating impliciti

La maggior parte dei casi studiati in letteratura sono rivolti alla raccomandazione in scenari in cui è presente un feedback esplicitamente espresso dagli utenti. Il *collaborative filtering* si è dimostrato promettenteanche in scenari in cui le preferenze degli utenti sono implicitamente ottenute da diversi tipi di feedback (ad esempio il numero delle volte che un utente ha ascoltato una canzone di un artista) [12]. Queste informazioni quantitative possono essere utilizzate come misura di preferenza e sopperire in un certo senso alla mancanza di un feedback esplicito. Tuttavia in alcuni scenari non è diponibile neanche questa informazione quantitativa, piuttosto si dispone solo di informazioni binarie in presenza di dati rilevanti (si pensi alle relazioni di amicizia in un social network, o la relazione di *follow* in un ambiente come *Twitter*).

In questo caso non ha senso parlare di preferenze negative: l'assenza di interazione non indica necessariamente una preferenza negativa verso quell'item. La maggior parte degli algoritmi presentati nella sezione precedente cercano di minimizzare una metrica di errore, in uno scenario in cui i rating sono impliciti, e qundi, non disponendo di una stima quantitativa del gradimento di un utente verso un certo item, ha poco senso utilizzare una metrica di errore. Un possibile approccio per gestire questo tipo di rating è quello di minimizzare una funzione obiettivo di ordinamento, piuttosto che il classico *Mean Square Error* della predizione; in entrambi i casi, comunque, il modello è tipicamente appreso minimizzando una funzione obiettivo. I metodi basati sull'ordinamento non cercano di prevedere la presenza o l'assenza di interazione tra utenti e item, ma cercano di modellare le decisioni degli utenti tra item.

2.3.1 CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering

In questo documento viene affrontato il problema della raccomandazione in scenari in cui si dispone solo di dati binari. La rappresentazione binaria utilizzata è piuttosto intuitiva: per ogni coppia *utente-item* è assegnato un *1* se è presente un'interazione, *0* altrimenti. Le interazioni vengono quindi interpretate come segnali positivi. L'algoritmo proposto apprende i parametri del modello massimizzando il *Mean Reciprocal Rank*, una metrica nota in letteratura nell'ambito della misurazione delle performance di *top-k* raccomandazioni. Introducendo un limite inferiore alla metrica *Mean Reciprocal Rank* si ottiene una complessità computazionale lineare. Sono stati condotti esperimenti su due dataset estratti da social network per dimostrare quanto l'algoritmo sia efficace e scalabile.

2.3.2 Alternating Least Squares for Personalized Ranking

L'articolo[81] presenta un algoritmo basato sull'ordinamento che lavora con rating impliciti che è basato su implicitALS[33] chiamato RankALS. Quest'ultimo riesce a ridurre la complessità della funzione obiettivo di *ranking* tramite operazioni matematiche che riescono a rendere l'algoritmo molto efficiente. Inoltre sia implicitALS che RankALS utilizzano il metodo di approssimazione di matrici chiamato *minimi quadrati alternati* o *alternating least squares* che permette di approssimare una matrice ricostruendola da matrici di grado inferiore. L'algoritmo riesce a migliorare ImplicitALS in tutti i casi di test.

Gli algoritmi saranno trattati dettagliatamente in seguito.

Articolo	Matrix Factorization	Graph Oriented	Annotation Based	Social CF	Social Group	Elicitation	Error Based	Rank Oriented
A Generalized Stochastic Block Model for Recommendation in SRN[37]	✓						✓	
A Semantic Model for Social RS[41]			✓				✓	
Augmenting collaborative recommender by fusing explicit social relationships[91]	✓	✓	✓	✓		✓		
Enhancing CF Systems with Personality Information[32]	✓						✓	
Interactive Recommendations in Social Endorsement Networks[46]			✓	✓			✓	
Recommendations in Taste Related Domains CF vs. SF[26]	✓			✓			✓	
Trustwalker[36] *		✓		✓			✓	
A hybrid approach to item recommendation in folksonomies[88]			✓				✓	
Pairwise interaction tensor factorization for personalized tag recommendation[66]	✓		✓				✓	
Tag recommendations in social bookmarking systems[38]			✓				✓	
Social Networking Feeds Recommending Items of Interest[22]				✓			✓	
A Social Network Based Recommender System[27]				✓			✓	

Tabella 2.1: Riepilogo dei documenti

Articolo	Matrix Factorization	Graph Oriented	Annotation Based	Social CF	Social Group	Elicitation	Error Based	Rank Oriented
Collaborative and Content-based Filtering for Item Recommendation on SBW[11]			✓				✓	
Factorization vs Regularization: Fusing Heterogeneous Social Relationships in Top-N Recommendation[90]	✓			✓	✓		✓	
Learning to Recommend with Social Trust Ensemble * [50]	✓			✓			✓	
Social Recommendation Using Probabilistic Matrix Factorization[51]	✓			✓			✓	
Wisdom of the Better Few: Cold Start Recommendation via Representative based Rating Elicitation[49]						✓	✓	
A Matrix Factorization Technique with Trust Propagation for Recommendation in SN[35]	✓			✓			✓	
Affiliation Recommendation using Auxiliary Networks[86]					✓		✓	
Social Network-based Recommendation: A Graph Random Walk Kernel Approach[47]		✓		✓			✓	
CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering[79] *	✓							✓
Alternating Least Squares for Personalized Ranking[81] *	✓							✓

Tabella 2.2: Riepilogo dei documenti (segue)

Capitolo 3

Algoritmi di riferimento

In questo capitolo verranno brevemente descritti gli algoritmi di riferimento della tesi, le scelte implementative effettuate e alcune considerazioni generali. I primi 4 sono algoritmi recenti: i primi due utilizzano le informazioni sociali per migliorare la qualità della raccomandazione. Il terzo e il quarto lavorano con rating impliciti e sono stati scelti per la loro qualità e perché sono in grado di lavorare con i dataset impliciti disponibili. I cinque algoritmi successivi, invece, sono algoritmi generici che vengono di solito presi in considerazione per il confronto. Learning To Recommend With Social Trust Ensemble, che sarà abbreviato con LTR, è stato di ispirazione per unificare la raccomandazione sociale e quella CF. Lo studio su Trustwalker ha permesso l'esplorazione delle potenzialità dei grafi. CLiMF e ALSPR hanno permesso il confronto con algoritmi che minimizzano funzioni di ordinamento ed impliciti. Gli altri sono algoritmi molto importanti in letteratura ai quali è necessario confrontarsi. Eccetto LTR e TrustWalker, gli altri algoritmi non utilizzano le informazioni sociali; LTR però minimizza l'RMSE e non una metrica di ordinamento, mentre TrustWalker minimizza la varianza dei risultati delle singole random walk. LTR e TrustWalker, inoltre, non sono in grado di lavorare con rating impliciti.

3.1 Learning to Recommend with Social Trust Ensemble

Questo algoritmo si è rivelato essere tra le soluzioni più promettenti nell'ambito della raccomandazione basata sulla fattorizzazione di matrici con informazioni sociali. Si è scelto quindi di implementarlo per poter effettuare dei confronti con la nostra proposta.

L'algoritmo riceve in ingresso la *matrice dei rating* (ovvero una matrice sparsa *user-item* in cui ogni cella contiene il rating che l'utente ha assegnato ad un item)

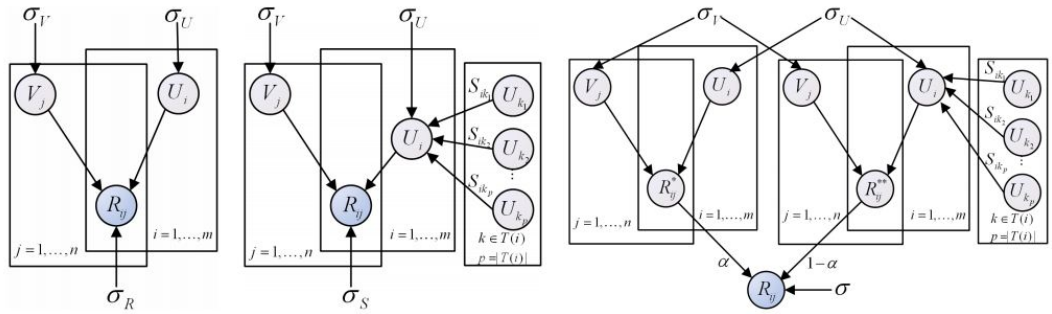


Figura 3.1: Processo di raccomandazione nell'algoritmo Learning to Recommend with Social Trust Ensemble

e la *rete di amicizie*, rappresentata come una matrice binaria *user-user* in cui un *1* indica la presenza della relazione di amicizia tra due utenti. Per apprendere le caratteristiche degli utenti viene utilizzata una tecnica di fattorizzazione sulla *matrice dei rating*, così da ottenere una rappresentazione degli utenti e degli item in uno spazio di features di dimensione k, dove k è il numero di features (le simulazioni sui dataset sono condotte con $k = 10$ e $k = 20$). Utilizzando una tecnica di apprendimento basata sulla *discesa del gradiente* si ottengono le matrici dei fattori latenti degli utenti e degli item, rispettivamente chiamate U e V. I vettori U_i e V_j rappresentano rispettivamente i vettori di features di dimensione k dell'utente u_i e dell'item v_j. Come già detto questo algoritmo si basa sull'assunzione che le scelte all'interno di una rete sociale siano influenzate dai gusti degli amici, questo processo sociale viene generalizzato dalla formula:

$$\hat{r}_{i,k} = \frac{\sum_{j \epsilon TU_{(i)}} R_{j,k} S_{i,j}}{|TU(i)|} \tag{3.1}$$

Dove $r_{i,k}$ è la predizione del rating che l'utente u_i darebbe all'item v_j, $e_{j,k}$ è il rating che l'utente u_j ha dato all'item v_k e $TU(i)$ è il set degli amici dell'utente u_i, e quindi $|TU(i)|$ è la cardinalità dell'insieme. Il gusto dell'utente e quello degli amici sono mediati da un parametro α che ha lo scopo di fondere le due componenti del processo di raccomandazione. Il parametro controlla quanto un utente sia propenso ad ascoltare il giudizio degli amici piuttosto che basarsi sul proprio. La figura 3.1 mostra con precisione il ruolo del parametro α: il contributo della fattorizzazione della *matrice dei rating* (a), viene mediato col contributo degli amici (b), ottenendo così una predizione dipendente da entrambi i fattori (c). Si noti che se $\alpha = 1$ l'algoritmo utilizza solamente la componente di predizione ottenuta dalla fattorizzazione della *matrice dei rating*, e che analogamente con

$\alpha = 0$ si utilizza solo la componente sociale.

Utilizzando le informazioni sociali la funzione obiettivo della discesa del gradiente diventa quindi:

$$L(R,S,U,V) = \frac{1}{2}\sum_{i=1}^{m} sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(\alpha(U_i^T V_j + (1-\alpha)\sum_{k \epsilon T(i)} S_{ik} U_k^T V_j))^2 + \\ + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||U||_F^2) \tag{3.2}$$

Dove $g(x)$ è la *logistic function*:

$$g(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

Il procedimento della discesa del gradiente ad ogni epoca t aggiorna le matrici U e V secondo le equazioni

$$U_i^{t+1} = U_i^t - \gamma \frac{\partial L}{\partial U} \tag{3.4}$$

$$V_j^{t+1} = V_j^t - \gamma \frac{\partial L}{\partial U} \tag{3.5}$$

dove γ è il fattore di apprendimento dell'algoritmo. Per stimare l'efficacia dell'algoritmo, gli autori hanno condotto diversi esperimenti su un dataset estratto da *Epinions*. Le metriche utilizzate sono il *Mean Absolute Errore (MAE)* (5.1) e il *Root Mean Square Error (RMSE)* (5.2).

3.2 TrustWalker

Come già descritto in precedenza *TrustWalker*[36] è un algoritmo basato sui grafi ed in particolare sul concetto di *random walk* su un grafo.

Una *random walk* di lunghezza k su un grafo è un processo stocastico con variabili aleatorie $X_1, X_2, \ldots, X_k$ tale che X_1 è il nodo di partenza e X_{i+1} è un vertice scelto uniformemente tra i vicini di X_i. Nel caso specifico se vogliamo predire il rating $\hat{r}_{ui}$ il nodo di partenza sarà u e i suoi vicini saranno gli utenti che hanno una relazione di amicizia con u, cioè tali che $t_{i,i+1} = 1$. La tabella di seguito riassume la notazione utilizzata. É importante notare che ogni random walk restituisce un rating, tranne in casi di errore che verranno trattati in seguito.

Una singola random walk Iniziando dall'utente u_0, si esegue la random walk per predire il rating $r_{u_0 i}$. Ad ogni passo k della random walk ci si troverà in un

Notazione	Descrizione
$\phi_{u,i,k}$	probabilità di fermare la random walk per l'item i sul nodo u allo step k.
$X_{u,i,k}$	v.a. di essere al nodo v in k passi iniziando da u
$X_{u,i}$	v.a. di essere al nodo v in qualche passo iniziando da u
S_u	v.a. di selezionare un utente v tra i membri dell'insieme TU_u
$Y_{u,i}$	v.a. di selezionare l'item j tra gli item rated da u
$XY_{u,i}$	v.a. di fermarsi al nodo v e selezionare l'item j rated da v iniziando da u

Tabella 3.1: Notazione utilizzata in TrustWalker

certo nodo u. Se u *possiede* il rating r_{ui} la random walk viene fermata e viene restituito questo rating. Se u *non possiede* il rating r_{ui} si hanno due opzioni:

- Con probabilità $\phi_{u,i,k}$ la random walk si ferma e viene scelto un item j simile a i e viene restituito r_{uj}.

- Con probabilità $1 - \phi_{u,i,k}$ si sceglie un utente $v|t_{uv} = 1$ e si continua la random walk sull'utente v.

La probabilità di scegliere un utente v amico di u è uniformemente distribuita, cioè $P(S_u = v) = \frac{1}{|TU_u|}$. La probabilità, invece, di scegliere un item j simile ad i per un utente u è pesata con le similarità: $P(Y_{ui} = j) = \frac{sim(i,j)}{\sum_{l \in RI_u} sim(i,l)}$. Per definire le similarità tra item è necessario definire la loro correlazione. A questo scopo è stato usato il coefficiente di correlazione di Pearson:

$$corr(i,j) = \frac{\sum_{u \in UC_{i,j}} (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)}{\sqrt{\sum_{u \in UC_{i,j}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{u \in UC_{i,j}} (r_{uj} - \overline{r}_u)^2}} \tag{3.6}$$

dove $UC_{i,j}$ è l'insieme degli utenti che hanno espresso ratings sia per i che per j. Si definisce quindi la similarità come:

$$sim(i,j) = \frac{1}{1 + e^{-\frac{|UCi,j|}{2}}} \times corr(i,j) \tag{3.7}$$

Non resta che definire la probabilità $\phi_{u,i,k}$. Essa è correlata alle similarità degli items rated da u e all'item i. Inoltre dovrebbe essere più alta per i vicini diretti di u che per quelli lontani. Gli autori la definiscono come:

$$\phi_{u,i,k} = \max_{j \in RI_u} sim(i,j) \times \frac{1}{1 + e^{-\frac{k}{2}}} \tag{3.8}$$

Con queste definizioni esiste la possibilità che una random walk continui all'infinito, perciò gli autori hanno deciso di limitare il numero massimo di passi (k) a 6. Se k supera 6, quindi, la random walk è considerata *senza successo*.

Gli autori dell'articolo descrivono anche delle formule per calcolare $XY_{u,i}$, $X_{u,i,k}$ e $X_{u,i}$, poiché il rating calcolato dovrebbe essere la media dei risultati delle random walk, pesata con queste probabilità.

$$\hat{r}_{u,i} = \sum_{\{(v,j)|R_{vj}\}} P(XY_{u,i} = (v,j))r_{vj} \tag{3.9}$$

In realtà nell'implementazione gli autori hanno semplicemente effettuato una media dei valori delle random walk, a causa di problemi di performance.

Per quanto riguarda la terminazione dell'algoritmo è stata scelta una condizione sulla varianza dei risultati: dopo ogni random walk viene calcolata la varianza dei rating restituiti finora. Se la differenza in valore assoluto tra questa varianza e la varianza calcolata prima della random walk scende sotto un certo valore ϵ, l'algoritmo termina.

$$|\sigma_{i+1}^2 - \sigma_i^2| \leq \epsilon \tag{3.10}$$

É stato posto un limite (10000) al numero massimo di random walk senza successo. Se questo limite viene raggiunto, la coppia utente-item viene considerata *non coperta*.

Implementazione L'algoritmo è stato implementato inizialmente in linguaggio Java. Le ragioni di questa scelta riguardano l'uso di *Neo4J*[1], un database basato sui grafi. Dato che l'algoritmo è basato sui grafi, questa scelta sembrava la più sensata. In realtà le performance del database sono state scarse; le query continue al database rallentavano enormemente la simulazione. Si è deciso, quindi, di precaricare in memoria tutti i dati all'inizio della simulazione, così da non inficiare troppo sul tempo di esecuzione. Ma ciò ha risolto solo in parte il problema delle performance: la *Java Virtual Machine* si è rivelata non adatta a supportare la mole di lavoro intensivo richiesta dall'algoritmo; così si è ritenuto indispensabile riscrivere il codice in linguaggio C, senza utilizzare nessun database, ma utilizzando una libreria per le matrici adatta allo scopo. Dopo la riscrittura le performance sono migliorate enormemente, accorciando i tempi di esecuzione di svariati ordini di grandezza.

[1] neo4j.org

3.3 CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering

L'algoritmo proposto [79] è in grado di effettuare raccomandazioni ricevendo in ingresso una matrice *user-item* che indica la presenza (o meno) di interazioni degli utenti con determinati item. Tale matrice può essere pensata come una *matrice di rating* binaria.

La funzione di predizione utilizzata nel modello è la classica e ampiamente utilizzata:

$$f_{ij} = \langle U_i, V_j \rangle \tag{3.11}$$

Dove U_i indica il vettore dei fattori latenti (di dimensione d) dell'utente i, e V_j indica il vettore dei fattori latenti (di dimensione d) dell'item j.

L'idea alla base di *CLiMF* è quella di ottimizzare il *Mean Reciprocal Rank* di una lista ottenuta da una ricerca. Inoltre è necessario introdurre dei termini di regolarizzazione generalmente impiegati in letteratura per tenere sotto controllo la complessità del modello e per evitare il fenomeno di *overfitting*. La funzione obiettivo adottata da *CLiMF* è dunque:

$$\begin{aligned} F(U,V) = &\sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}[\ln g(U_i^T V_j) + \sum_{k=1}^{N} \ln(1 - Y_{ik} g(U_i^T V_k - U_i^T V_j))] + \\ &- \frac{\lambda}{2}(||U||^2 + ||V||^2) \end{aligned} \tag{3.12}$$

Viene utilizzato lo *stochastic gradient ascent* per massimizzare la funzione obiettivo (3.12), le derivate parziali rispetto ad U_i (3.13) e V_j (3.14) si calcolano come:

$$\begin{aligned} \frac{\partial F}{\partial U_i} = &\sum_{j=1}^{N} Y_{ij}[g(-f_{ij})V_j + \\ &+ \sum_{k=1}^{N} \frac{Y_{ik} g'(f_{ik} - f_{ij})}{1 - Y_{ik} g(f_{ik} - f_{ij})}(V_j - V_k)] - \lambda U_i \end{aligned} \tag{3.13}$$

$$\begin{aligned} \frac{\partial F}{\partial V_j} = &Y_{ij}[g(-f_{ij}) + \\ &+ \sum_{k=1}^{N} Y_{ik} g'(f_{ij} - f_{ik})(\frac{1}{1 - Y_{ik} g(f_{ik} - f_{ij})} + \\ &- \frac{1}{1 - Y_{ij} g(f_{ij} - f_{ik})})]U_i - \lambda V_j \end{aligned} \tag{3.14}$$

Dove $g(x)$ è la *funzione logistica* (3.3) e $g'(x)$ la sua derivata.

Le metriche utilizzate dagli autori dell'articolo sono Mean Reciprocal Rank (MRR), recall(5) e 1-call@5. Le prime due saranno descritte nelle sezioni 5.1.5 e 5.1.2 e saranno utilizzate per il confronto degli algoritmi. I risultati ottenuti dagli autori mostrano una MRR minima di 0.077 e una massima di 0.292; la recall minima riportata invece è di 0.022, mentre la massima è di 0.216. É importante notare come queste metriche siano fortemente dipendenti dal dataset, dalla metodologia di testing e dai parametri dell'algoritmo, i quali richiedono un'attenta calibrazione.

3.4 Alternating Least Squares for Personalized Ranking

Come già descritto prima, l'algoritmo *Alternating Least Squares for Personalized Ranking* (ALSPR)[81] riceve in input una matrice di rating binaria. Esso minimizza la funzione di *ranking* definita dalla (3.15).

$$f_R(\Theta) = \sum_{u \in U} \sum_{i \in I} c_{ui} \sum_{j \in I} s_j [(\hat{r}_{ui} - \hat{r}_{uj}) - (r_{ui} - r_{uj})]^2 \tag{3.15}$$

dove c_{ui} e s_j sono parametri della funzione e sono funzione di Θ. Questa funzione è stata introdotta nella letteratura dei sistemi di raccomandazione in [34]. Utilizzando dei rating impliciti, $c_{ui} = 0$ se $r_{ui} = 0$ e 1 altrimenti. Il ruolo di c_{ui} è quindi quello di selezionare le coppie utente-item corrispondenti a rating impliciti. Il ruolo di s_j, invece è quello di modellare l'importanza di un item; abbiamo due versioni diverse dell'algoritmo: *support weighted* e *non support based*; nella prima versione $s_j = |U_j|$, cioè corrisponde al numero di utenti che hanno dato un rating all'item j; nella seconda versione $s_j = 1 \quad \forall j$.

L'algoritmo quindi cerca di minimizzare questa funzione di ranking con il metodo dei minimi quadrati alternati. L'implementazione suggerita dagli autori è molto efficiente, avendo complessità $O(TF^2 + (U + I)F^3)$, dove T è la cardinalità dei rating, U quella degli utenti, I quella degli item ed F il numero di fattori

latenti. Le derivate della funzione di ranking sono descritte nelle (3.16) e (3.17).

$$
\begin{aligned}
\frac{\partial f_R(P,Q)}{\partial p_u} &= \sum_{i\in I} c_{ui} \sum_{j\in I} s_j \left[(q_i - q_j)^T p_u - (r_{ui} - r_u j) \right] (q_i - q_j) = \\
&= \underbrace{\left(\sum_{j\in I} s_j \right)}_{\tilde{1}} \underbrace{\left(\sum_{i\in I} c_{ui} q_i q_i^T \right)}_{\overline{A}} p_u - \underbrace{\left(\sum_{i\in I} c_{ui} q_i \right)}_{\overline{q}} \underbrace{\left(\sum_{j\in I} s_j q_j^T \right)}_{\tilde{q}^T} p_u + \\
&\quad - \underbrace{\left(\sum_{j\in I} s_j q_j \right)}_{\tilde{q}} \underbrace{\left(\sum_{i\in I} c_{ui} q_i^T \right)}_{\overline{q}^T} p_u + \underbrace{\left(\sum_{i\in I} c_{ui} \right)}_{\overline{1}} \underbrace{\left(\sum_{j\in I} s_j q_j q_j^T \right)}_{\tilde{A}} p_u + \\
&\quad - \underbrace{\left(\sum_{i\in I} c_{ui} q_i r_{ui} \right)}_{\overline{b}} \underbrace{\left(\sum_{j\in I} s_j \right)}_{\tilde{1}} + \underbrace{\left(\sum_{i\in I} c_{ui} q_i \right)}_{\overline{q}} \underbrace{\left(\sum_{j\in I} s_j r_{uj} \right)}_{\tilde{r}} + \\
&\quad + \underbrace{\left(\sum_{i\in I} c_{ui} r_{ui} \right)}_{\overline{r}} \underbrace{\left(\sum_{j\in I} s_j q_j \right)}_{\tilde{q}} - \underbrace{\left(\sum_{i\in I} c_{ui} \right)}_{\overline{1}} \underbrace{\left(\sum_{j\in I} s_j q_j r_{uj} \right)}_{\tilde{b}} = \\
&= \left(\tilde{1}\overline{A} - \overline{q}\tilde{q}^T - \tilde{q}\overline{q}^T + \overline{1}\tilde{A} \right) p_u - \left(\overline{b}\tilde{1} - \overline{q}\tilde{r} - \overline{r}\tilde{q} + \overline{1}\tilde{b} \right)
\end{aligned}
\tag{3.16}
$$

$$
\begin{aligned}
\frac{\partial f_R(P,Q)}{\partial q_i} &= \\
&= \sum_{u\in U}\sum_{j\in I} c_{ui}s_j p_u \left[p_u^T(q_i - q_j) - (r_{ui} - r_{uj})\right] + \\
&\quad + \sum_{u\in U}\sum_{j\in I} c_{uj}s_i(-p_u)\left[p_u^T(q_j - q_i) - (r_{uj} - r_{ui})\right] = \\
&= \underbrace{\left(\sum_{u\in U} c_{ui}p_u p_u^T\right)}_{\overline{A}}\underbrace{\left(\sum_{j\in I} s_j\right)}_{\tilde{1}} q_i - \underbrace{\left(\sum_{u\in U} c_{ui}p_u p_u^T\right)}_{\overline{A}}\underbrace{\left(\sum_{j\in I} s_j q_j\right)}_{\tilde{q}} + \\
&\quad - \underbrace{\left(\sum_{u\in U} c_{ui}p_u r_{ui}\right)}_{\overline{b}}\underbrace{\left(\sum_{j\in I} s_j\right)}_{\tilde{1}} + \underbrace{\left(\sum_{u\in U} c_{ui}\overbrace{\left(\sum_{j\in I} s_j r_{uj}\right)}^{\tilde{r}} p_u\right)}_{\overline{\overline{p}}_1} + \\
&\quad - \underbrace{\left(\sum_{u\in U} p_u p_u^T\overbrace{\left(\sum_{j\in I} c_{uj}q_j\right)}^{\overline{q}}\right)}_{\overline{\overline{p}}_2} s_i + \underbrace{\left(\sum_{u\in U} p_u p_u^T\overbrace{\left(\sum_{j\in I} c_{uj}\right)}^{\overline{1}}\right)}_{\overline{\overline{A}}} s_i q_i + \\
&\quad + \underbrace{\left(\sum_{u\in U}\overbrace{\left(\sum_{j\in I} c_{uj}r_{uj}\right)}^{\overline{q}} p_u\right)}_{\overline{\overline{p}}_3} s_i - \underbrace{\left(\sum_{u\in U} p_u r_{ui}\overbrace{\left(\sum_{j\in I} c_{uj}\right)}^{\overline{1}}\right)}_{\overline{\overline{b}}} s_i = \\
&= \left(\overline{A}\tilde{1} + \overline{\overline{A}}s_i\right) q_i + \left(-\overline{A}\tilde{q} - \overline{b}\tilde{1} + \overline{\overline{p}}_1 - \overline{\overline{p}}_2 s_i + \overline{\overline{p}}_3 s_i - \overline{\overline{b}}_2 s_i\right)
\end{aligned}
\tag{3.17}
$$

Le metriche utilizzate dagli autori dell'articolo sono: Error rate, Average Rating Position(ARP) e recall. Le ultime due saranno discusse rispettivamente nelle sezioni 5.1.6 e 5.1.2.

I risultati ottenuti dagli autori in termini di ARP sono 0.022 nel caso migliore e 0.13 nel caso peggiore. La recall(50) massima ottenuta dagli autori è 0.372,

mentre la minima è 0.1331. Anche in questo caso i risultati variano in base al dataset utilizzato, dalla metodologia di testing e dai parametri utilizzati, i quali richiedono delle operazioni di calibrazione dell'algoritmo.

Inizialmente l'algoritmo è stato implementato in Matlab dato l'elevato numero di operazioni matriciali, ma per motivi di performance si è implementato anche in C. Implementare l'algoritmo in C ha richiesto uno sforzo maggiore, che però è stato ricompensato da una velocità di esecuzione molto maggiore. Inoltre l'implementazione in C ha permesso di fare delle piccole ottimizzazioni che evitavano il ricalcolo di alcuni fattori. L'articolo richiede anche l'*inversione di matrice* che è stata implementata con il metodo di eliminazione di *Gauss Jordan* che è uno tra i più veloci. Successivamente verrà mostrato come l'algoritmo possa essere modificato per incorporare le informazioni sociali.

3.5 Asymmetric SVD

L'Asymmetric SVD è un algoritmo collaborativo di tipo item-based sviluppato da Yehuda Koren e vincitore della competizione Netflix Prize[43]. Esso utilizza la fattorizzazione delle matrici con fattori latenti: viene effettuata una discesa del gradiente che minimizza l'RMSE dei rating predetti; i rating per ogni item i non presente nel profilo dell'utente u vengono calcolati con la (3.18).

$$r_{ui} = b_{ui} + q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j \right) \tag{3.18}$$

La prima parte della formula è una stima con $b_{ui} = \mu + b_u + b_i$ dove μ rappresenta la media globale dei rating del dataset, b_u e b_i sono le distanze dalla media totale μ rispettivamente della media dei ratings dati dall'utente u e della media dei ratings ricevuti dall'item i. $|R(u)|$ è la cardinalità dell'insieme dei rating espressi dall'utente u. q_i e x_j sono i vettori che contengono i fattori latenti degli item i e j e vengono calcolati con la (3.19).

$$\begin{aligned} \min_{q,x,b} \sum_{(u,i) \in \kappa} & \left(r_{ui} - \mu - b_u - b_i - q_i^T \cdot \left(|R(u)|^{-\frac{1}{2}} \cdot \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot x_j \right) \right)^2 + \\ & + \lambda \cdot \left(b_u^2 + b_i^2 + \left\| q_i^T \right\|^2 + \sum_{j \in R(u)} \|x_j\|^2 \right) \end{aligned} \tag{3.19}$$

dove κ rappresenta un insieme casuale di coppie $\langle utente, item \rangle$ che è usato per calcolare l'errore tra i rating predetti e quelli del training set. Il secondo addendo è un fattore di regolarizzazione che penalizza valori alti per i parametri.

3.6 Non-normalized Cosine K Nearest NeighBour (NN-CosNgbr)

L'algoritmo Item-Item Cosine KNN (brevemente *COS*) appartiene alla classe di algoritmi *item-based*. Nella fase di apprendimento l'algoritmo riceve in ingresso la URM di dimensioni $m \times n$ e restituisce in output una matrice $M \in \mathbb{R}^{m \times n}$ di similarità tra item. Gli item sono pensati come vettori nello spazio degli utenti di lunghezza m. La similarità tra due item viene calcolata come il coseno tra i loro vettori associati, secondo la (3.20).

$$sim\,(i,j) = cos\left(\vec{i},\vec{j}\right) = \frac{\vec{i}\cdot\vec{j}}{\left\|\vec{i}\right\|^2 \times \left\|\vec{j}\right\|^2} \tag{3.20}$$

Ogni cella $m_{i,j}$ del modello M rappresenta quindi la similarità tra i e j che varia tra 1 (massima similarità) e 0 (minima similarità). La fase di raccomandazione per un utente u consiste nell'effettuare il prodotto tra il *profilo utente* p_u e la matrice M. In questo modo per ogni item i non presente nel profilo utente verrà assegnato un punteggio in base al suo coefficiente di similarità con ogni item $j_{1,\dots,n}$ che invece è presente. Questo punteggio può essere considerato come il punteggio che l'utente assegnerebbe a quell'item. Viene così generata una lista ordinata in modo decrescente di rating predetti, della quale vengono presi i primi K elementi: questa è la lista di item raccomandati all'utente.

L'eliminazione dei $n - K$ item si basa sul principio che questi item danno un contributo di similarità minimo e possono essere considerati rumore. Il parametro K può dipendere dal dataset: se troppo grande, viene inserito più rumore nella raccomandazione ma potrebbe permettere di suggerire item meno ovvi; se troppo piccolo elimina il rumore ma potrebbe portare alla perdita di item rilevanti.

3.7 MovieAVG

MovieAVG è un algoritmo non personalizzato che raccomanda gli item con la media di rating più alta. Il rating predetto dell'utente u è quindi la media dei rating che la comunità ha espresso sull'item i, indipendentemente dai rating espressi da u.

3.8 Pure SVD

PureSVD è un algoritmo collaborativo che utilizza i fattori latenti e che si è mostrato ottimo nel garantire raccomandazioni di alta qualità in termini di *recall*[17]. Dato un numero di fattori latenti f, l'SVD permette di fattorizzare la URM in tre matrici f-dimensionali: $\mathbf{U}$, $\mathbf{\Sigma}$ e $\mathbf{Q}$. Esso calcola l'approssimazione di rango f della matrice con la (3.21),

$$R_f = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{Q}^T \tag{3.21}$$

dove $\mathbf{U}$ e $\mathbf{Q}$ sono matrici ortonormali e $\mathbf{\Sigma}$ è diagonale. L'utente u e l'item i sono rappresentati da vettori f-dimensionali p_u e q_i. La predizione del rating è calcolata con la (3.22).

$$\hat{r}_{ui} = p_u \cdot q_i^T \tag{3.22}$$

Se definiamo $\mathbf{P} = \mathbf{\Sigma} \cdot \mathbf{U}$ possiamo osservare che, dato che $\mathbf{U}$ e $\mathbf{Q}$ sono matrici ortonormali, $\mathbf{P} = \mathbf{R} \cdot \mathbf{Q}$ e la (3.22) può essere riscritta come la (3.23).

$$\hat{r}_{ui} = r_u \cdot Q \cdot q_i^T \tag{3.23}$$

Possiamo infine osservare che il prodotto interno $q_i \cdot q_j^T$ rappresenta la similarità tra gli item i e j.

3.9 Top Rated

TopRated è un algoritmo non personalizzato che raccomanda gli item più popolari del dataset, ovvero gli item col più alto numero di rating. Si è deciso di includerlo nei test poiché mostra come alcuni metodi di test tendano fortemente a favorire algoritmi che raccomandano item popolari.

É da notare come in questo caso i rating dell'utente u non possano essere inferiti, ma l'output dell'algoritmo è una lista ordinata di item; ciò fa sì che l'RMSE o altre metriche di errore non possano essere applicate.

Capitolo 4

Dataset

4.1 Natura e dominio dei dati

Gli studi che andremo ad illustrare in seguito, sono stati condotti su tre dataset reali reperibili online: *Last.fm*[1], *Delicious*[2] ed *Epinions*[3].

Last.fm è un sito di musica inglese fondato nel 2002, recentemente acquisito dalla *CBS Interactive*. Utilizzando un sistema di raccomandazione per la musica (*audioscrobbler*), *Last.fm* è in grado di costruire, per ogni utente, un dettagliato profilo musicale basato su informazioni raccolte dai dettagli delle canzoni ascoltate dall'utente (sia da stazioni radio online, sia da canzoni riprodotte su altri dispositivi). *Last.fm* offre diverese caratteristiche tipiche di un social network e può raccomandare canzoni ed artisti simili ai gusti dell'utente.

Delicious (un tempo conosciuto come *del.icio.us*) è un social network di *bookmarking* pensato per immagazzinare, condividere e scoprire riferimenti a pagine web. Il sito è stato fondato da Joshua Schachter nel 2003, per poi essere acquisito da *Yahoo!* nel 2005. *Delicious* usa un sistema di classificazione in cui ogni utente può assegnare *tag* ad ogni *bookmark* (generando una sorta di *folksonomia*). Inoltre ogni utente può raggrupare i propri link per argomento, annotando informazioni quali il titolo e la descrizione della pagina.

Epinions è un sito di recensioni fondato nel 1999 e successivmanete acquisito da *Ebay* nel 2005. Ogni visitatore può leggere recensioni scritte da altri utenti su una grande varietà di *item* (film, libri, prodotti di vario genere, ...). Gli utenti hanno la possibilità di esprimere il proprio parere sulle recensioni utilizzando una scala da uno a cinque (*Very Helpful*, *Helpful*, *Somewhat Helpful*, *Not Helpful*, *Off Topic*).

[1] http://www.last.fm
[2] http://www.grouplens.org/node/462#attachments
[3] http://www.trustlet.org/wiki/Epinions_dataset

4.2 last.fm

Il dataset estratto da *Last.fm* è così composto:

- 1.892 utenti
- 17.632 artisti
- 12.717 relazioni bidirezionali di amicizia tra utenti
- 92.834 relazioni di *listen* tra utenti e artisti
- 11.946 *tag*
- 186.479 assegnamenti di *tag* da un utente verso un artista (ossia tuple *[utente, tag, artista]*)

Gli oggetti del nostro interesse sono gli utenti e gli artisti, questi ultimi saranno considerati come *item* e quindi oggetto della raccomandazione. Di conseguenza sono state prese in considerazione le 92.834 relazioni di *listen* e le 12.717 relazioni di amicizia che intercorrono tra gli utenti. Sebbene si tratti di informazioni utili per altri tipi di raccomandazione (ad esempio *annotation based*) i *tag* e le rispettive relazioni non sono stati tenuti in considerazione. É importante notare che per ogni utente è stata fornita solo la top 50 degli artisti più ascoltati.

4.2.1 Rappresentazione dei Dati

Le relazioni di amicizia sono state raccolte in una matrice binaria e simmetrica (*trust matrix*) dove un 1 indica la presenza di una relazione di amicizia tra due utenti (viceversa uno 0 l'assenza).

Il dataset in questione non dispone di rating espliciti, cioè di una rappresentazione quantitativa in una scala ben definita che indichi quanto un utente abbia apprezzato un item. L'unica informazione quantitativa di cui si dispone è il numero di volte in cui un artista è stato ascoltato da un utente. Si è scelto di ovviare il problema in due modi:

- ignorando l'informazione quantitativa e realizzando una *rating matrix* binaria dalle relazioni di listen, ponendo 1 in presenza di una relazione e 0 altrimenti
- utilizzando una funzione nota in letteratura [61] per ottenere dei rating in una scala da $[1, 5]$

Sono state costruite quindi due *rating matrix*: una di rating impliciti e una di rating espliciti ottenuti in funzione del conteggio dei *listen* di ogni utente. I rating sono così ottenuti [61] :

$$r_{i,j} = 4 \cdot \left(1 - \sum_{k'=1}^{k-1} freq_{k'}(i)\right) + 1 \tag{4.1}$$

Dove $freq_k(i)$ indica la frequenza del k-esimo artista più ascoltato dall'utente.

4.2.2 Statistiche

Dal grafico in figura 4.1 si evince che utente ha mediamente circa **27** amici, e che solo il 20% degli utenti ha meno 5 amici.

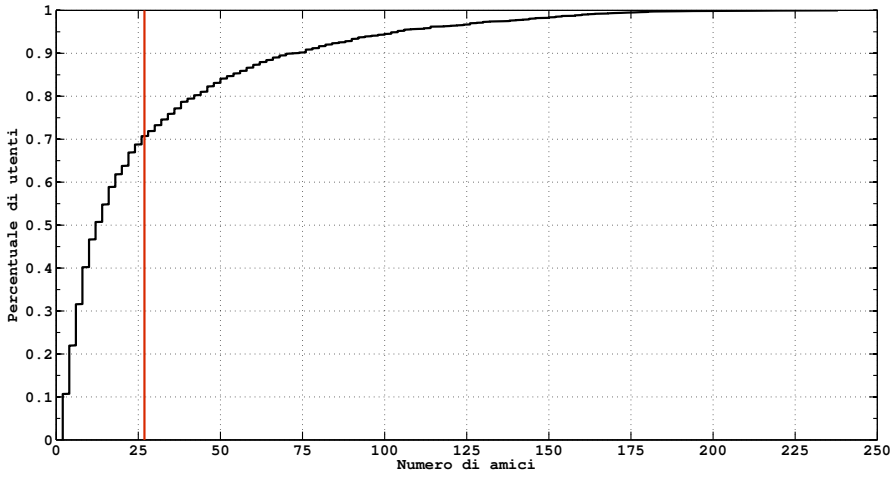


Figura 4.1: LastFM friend per User

Bisogna tenere in particolare considerazione che il dataset originale è stato ridotto dagli autori per cui la distribuzione dei rating per utente in Figura 4.2 presenta due caratteristiche importanti:

- ogni utente ha al massimo **50** rating
- appena il 3% degli utenti ha meno di **45** rating, per cui la percentuale di utenti *cold start* è irrisoria

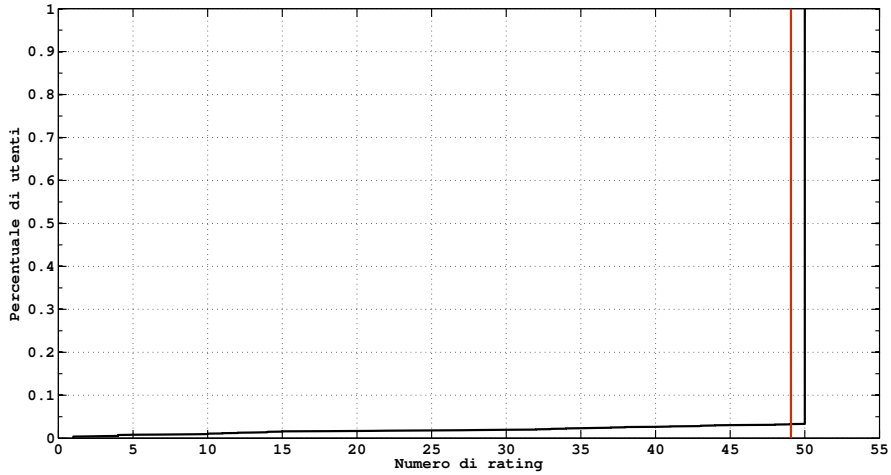


Figura 4.2: LastFM rating per User

Si noti che la distribuzione dei rating per utente in Figura 4.2 vale per entrambe le rating matrix, poiché sebbene i valori delle due matrici differiscano, la densità è la stessa. Infatti la *matrice dei rating impliciti* può teoricamente essere ottenuta direttamente dalla *matrice dei rating espliciti* ponendo 1 in corrispondenza di ogni valore nell'intervallo $[1, 5]$.

Il grafico in Figura 4.3 mostra come si distribuiscono i valori dei rating ottenuti dalla (4.1).

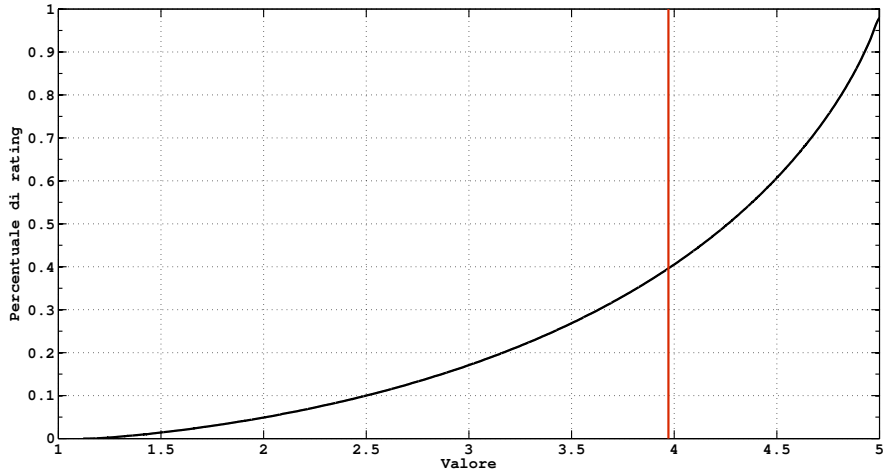


Figura 4.3: LastFM distribuzione dei rating

4.3 Delicious

Il dataset estratto da *Delicious* è così composto:

- 1.867 utenti
- 69.226 url
- 38.581 url principali (ad esempio *www.delicious.com* per *http://www.delicious.com, http://www.delicious.com/tag, http://www.delicious.com/help/api*, in figura 4.4 sono mostrati i domini più popolari)
- 53.388 tag
- 7.668 relazioni bidirezionali di amicizia tra utenti
- 437.593 relazioni di *assegnamento di tag*
- 104.799 relazioni di *bookmark* tra utenti e url, ottenute dalle relazioni di assegnamento dei tag (ossia tuple *[utente, tag, url]*)

Gli oggetti del nostro interesse sono gli utenti e gli url, questi ultimi saranno considerati come *item* e quindi oggetto della raccomandazione. Di conseguenza sono state prese in considerazione le 104.799 relazioni di *bookmark* e le 7.668 relazioni di amicizia che intercorrono tra gli utenti. Le 437.593 relazione di assegnamento dei tag, come spiegheremo a breve, sono state utilizzate per stimare quanto un utente sia interessato ad un particolare url.

Figura 4.4: Tag cloud per i domini di delicious

4.3.1 Rappresentazione dei Dati

Le relazioni di amicizia sono state raccolte in una matrice binaria e simmetrica (*trust matrix*) dove un 1 indica la presenza di una relazione di amicizia tra due utenti (viceversa uno 0 l'assenza).

Il dataset in questione non dispone di rating espliciti, cioè di una rappresentazione quantitativa in una scala ben definita che indichi quanto un utente abbia apprezzato un item. L'unica informazione quantitativa di cui si dispone è il numero di tag assegnate ad un url da un utente. Si è scelto di ovviare il problema in due modi:

- ignorando l'informazione quantitativa e realizzando una *rating matrix* binaria dalle relazioni di assegnamento di tag, ponendo 1 in presenza di una relazione e 0 altrimenti

- adattando una funzione nota in letteratura [15] per ottenere dei rating in una scala da $[1, 5]$

Sono state costruite quindi due *rating matrix*: una di rating impliciti e una di rating espliciti ottenuti in funzione del conteggio dei tag assegnati da ogni utente ad un particolare url. I rating sono così ottenuti [15] :

$$r_{u,i} = round\ up\left(5 \cdot \frac{AP(u,i)}{\max_{c \in U}(AP(c,i))}\right) \tag{4.2}$$

Dove U è l'insieme degli utenti che hanno assegnato un tag all'item i, *round up* arrotonda per eccesso, e la funzione AP è così definita:

$$AP(u,i) = \ln\left(\frac{numero\ di\ assegnamenti\ di\ tag\ dell'utente\ u\ per\ l'item\ i}{numero\ di\ tag\ assegnate\ dall'utente\ u} + 1\right) \tag{4.3}$$

4.3.2 Statistiche

Dal grafico in figura 4.5 si evince che ogni utente ha mediamente circa **16** amici, e che solo il 15% degli utenti ha meno di 5 amici.

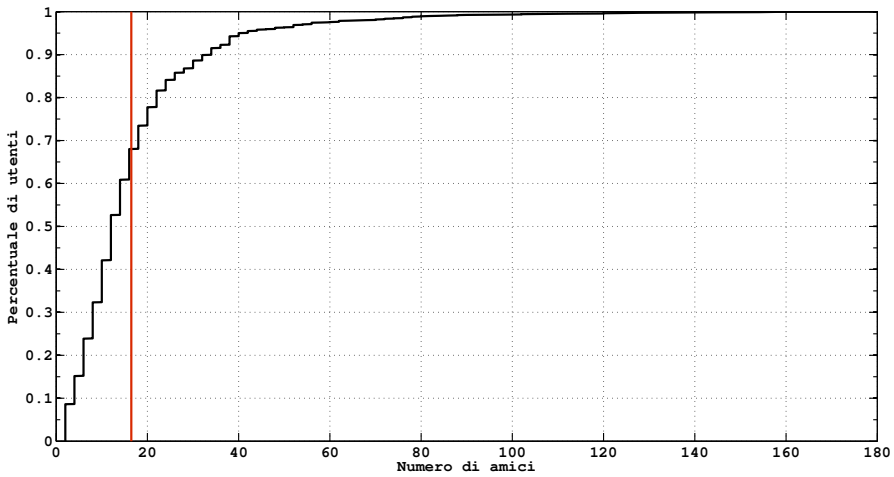


Figura 4.5: delicious: amici per utente

Dal grafico in figura 4.6 si evince che ogni utente ha in media **56** amici e che solo il 5% degli utenti ha meno di 5 rating.

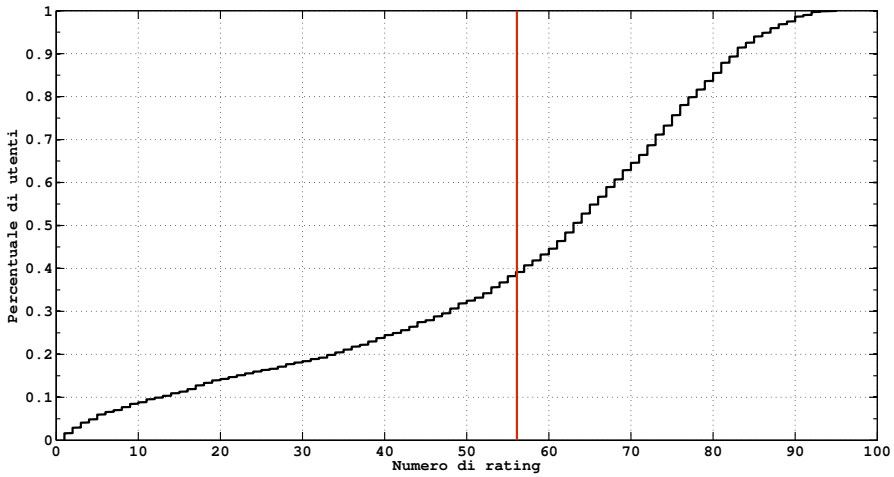


Figura 4.6: delicious: rating per utente

Si noti che la distribuzione dei rating per utente in figura 4.6 vale per entrambe le rating matrix, poiché sebbene i valori delle due matrici differiscano, la densità è la stessa. Infatti la *matrice dei rating impliciti* può teoricamente essere ottenuta

direttamente dalla *matrice dei rating espliciti* ponendo 1 in corrispondenza di ogni valore nell'intervallo $[1, 5]$.

Il grafico in figura 4.7 mostra come si distribuiscono i valori dei rating ottenuti dalla (4.2).

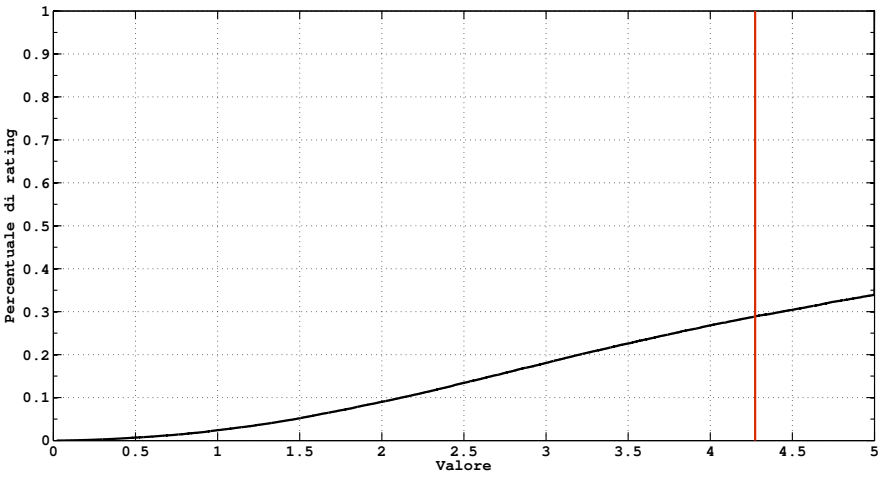


Figura 4.7: delicious: distribuzione dei rating

4.4 Epinions

Il dataset estratto da *Epinions* è così composto:

- 49.290 utenti
- 139.738 item
- 664.824 recensioni
- 487.181 relazioni bidirezionali di amicizia tra utenti

Per questioni legate alle performance è stato effettuato un sottocampionamento al 10% su tutti gli utenti.

4.4.1 Rappresentazione dei Dati

É stata costruita una matrice dei rating utilizzando le recensioni, queste infatti sono una misura esplicita del gradimento espresso da un utente in favore di un item. Si tratta di rating discreti nell'intervallo $[1, 5]$. É importante notare che, mentre dei due dataset precedenti si dispone direttamente di una versione binaria della URM, questo dataset deve esser reso binario. Il procedimento seguito per rendere il dataset binario considera solo i rating espliciti pari a 5.

Le relazioni di amicizia sono state raccolte in una matrice binaria e simmetrica (*trust matrix*) dove un 1 indica la presenza di una relazione di amicizia tra due utenti (viceversa uno 0 l'assenza).

4.4.2 Statistiche

Dal grafico in figura 4.8 si evince che utente ha mediamente circa **20** amici, e che ben il 55% degli utenti ha meno 5 amici.

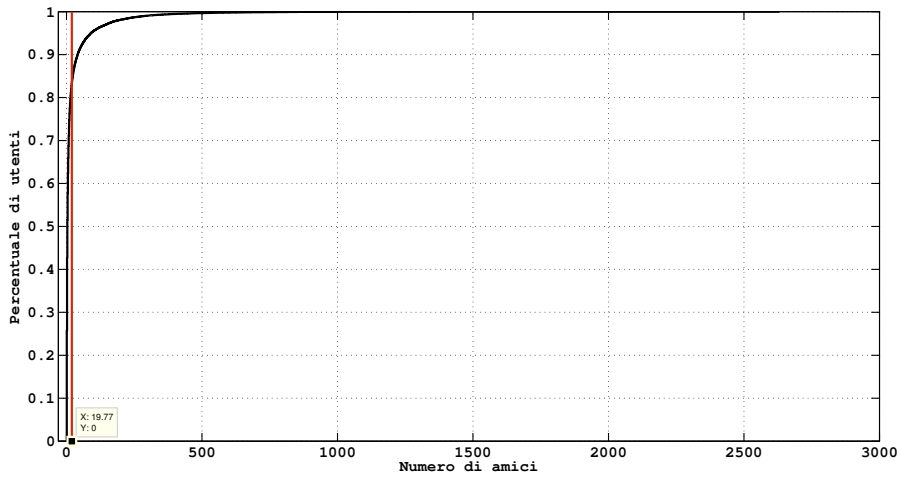


Figura 4.8: epinions: amici per utente

Il grafico in figura4.9 si evince che in media un utente ha espresso un parere su circa **13** item, e che ben il 52% degli utenti ha espresso meno di 5 preferenze.

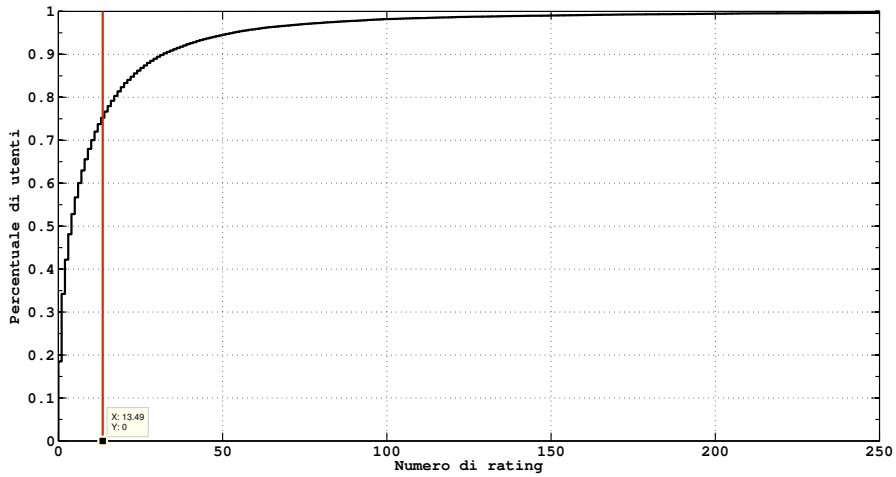


Figura 4.9: epinions rating per utente

Dal grafico in figura 4.10 mostra come si distribuiscono i valori dei rating.

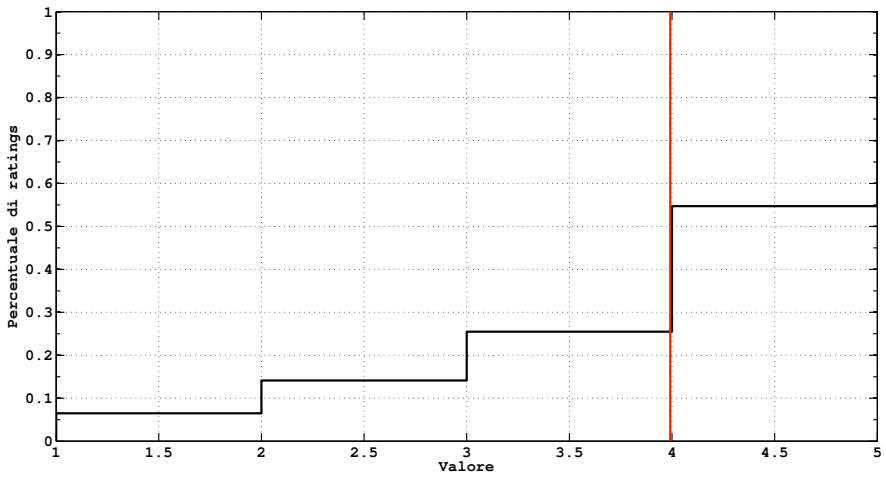


Figura 4.10: epinions: rating distribution

4.5 Preparazione dei dati

É necessario predisporre i dati a disposizione affinché questi siano processabili dagli algoritmi studiati. Come illustrato nelle sezioni precedenti, i dati ottenuti da *lastfm* e *delicious* sono stati processati in modo da ottenere una rating matrix con valori reali in $[1, 5]$ e una rating matrix binaria. I dati di *Epinions* sono già presentati sotto forma di matrice dei rating con valori discreti in $[1, 5]$. Tale matrice è stata quindi processata in modo da ottenere una matrice dei rating binaria con (1 in presenza di rating uguali a 5 e 0 altrimenti). Per ogni dataset si dispone quindi di una matrice binaria e di una matrice di rating. Distinguiamo gli algoritmi esaminati in due categorie non disgiunte:

- algoritmi in grado di processare rating con valori nell'intervallo $[1, 5]$ (rating espliciti);
- algoritmi in grado di processare rating binari;

Gli algoritmi *PureSVD*, *COS* e *TopRated* funzionano correttamente con entrambi gli input.

4.6 Neo4J

Neo4j è un database ad alte prestazioni di tipo NOSQL basato sui *grafi*. Esso permette di lavorare con una struttura a *grafo* orientata agli oggetti e flessibile,

invece che con tabelle statiche, beneficiando comunque di caratteristiche avanzate come le *transazioni*.

Neo4j fornisce un modello intuitivo per la rappresentazione dei dati, un gestore dello storage ottimizzato per memorizzare strutture a grafi per ottenere massime performance e scalabilità. Fornisce anche un *framework* per l'attraversamento dei nodi ad alta velocità e offre delle *API* per interagire con il linguaggio *Java*.

I *grafi* sono la struttura dati più generica, capace di rappresentare ogni tipo di dato in maniera facilmente accessibile. Si è scelta questa struttura poichè i dataset disponibili si prestano bene ad essere rappresentati su un grafo; LastFM ad esempio dispone di tre tipi di entità: *utenti*, *artisti* e *tag*, mentre Delicious dispone di *utenti*, *bookmark* e *tag*. Si è scelto quindi di modellare queste entità come *nodi* di un grafo. Ad ogni nodo è associato un certo numero di *proprietà*, come l'id, il tipo e altre proprietà variabili in base al dataset: il BOOKMARK ad esempio possiede la proprietà URL.

Ognuna delle relazioni ternarie che legano i tre tipi di entità è stata trasformata in più relazioni binarie: UTENTE-TAG-BOOKMARK nel caso di delicious è stata trasformata nelle relazioni UTENTE-TAG, UTENTE-BOOKMARK e TAG-BOOKMARK. Neo4j offre anche la possibilità di associare delle proprietà alle relazioni: la relazione UTENTE-TAG possiede la proprietà bookmark che contiene l'id del bookmark associato. Questo non è il modo più efficiente per memorizzare una relazione di questo tipo, ma garantisce, di contro, una grande semplicità di utilizzo. Per l'accesso ai nodi, in fase di creazione del database, è stato associato un indice di tipo stringa ad ogni nodo: ad esempio il bookmark numero 165 aveva associato un id B165 in modo tale da esser chiara la distinzione tra quest'ultimo e l'utente o l'item numero 165.

Le informazioni sociali, come l'amicizia, sono state modellate semplicemente come relazioni binarie tra nodi utente. I rating espliciti sono stati modellati anch'essi come relazioni binarie tra utenti e item.

L'accesso al database avveniva tramite API Java. Vi sono diversi modi di accedere al database: il primo, orientato agli oggetti, permette di accedere al nodo tramite la sua chiave primaria, ovvero un id associato dal database. Il secondo metodo permette di eseguire delle *query* al database ed eseguire il *fetching* del risultato della query su oggetti di tipo NODE o RELATIONSHIP, che permettono di navigare la struttura a grafo.

Il linguaggio di query nativo è Cypher, che ha costrutti simili al SQL con l'aggiunta di costrutti relativi soltanto ai grafi: il costrutto START, ad esempio, permette di specificare il nodo di partenza, il costrutto MATCH permette di spe-

cificare i nodi associati a quello di partenza specificando opzionalmente anche il tipo di relazione. Neo4j supporta anche altri tipi di linguaggi come *Gremlin*, anch'esso molto utilizzato nel campo dei database basati sui grafi. É stato scelto Cypher per la sua semplicità unita alla potenza espressiva e alla sua similarità con SQL.

Alcune operazioni come l'attraversamento del grafo, relative soprattutto all'algoritmo TrustWalker[36], sono state implementate con la funzione Traversal, particolarmente adatta ed ottimizzata per navigare un grafo a partire da un nodo.

Java e Neo4j, sebbene semplici da usare, non si sono rivelati particolarmente adatti ad eseguire la mole di calcolo richiesta dagli algoritmi. Così sì è deciso di implementare la fase di apprendimento del modello e le operazioni di testing degli algoritmi in linguaggio C, utilizzando una libreria proprietaria per le matrici sparse; questa scelta ha permesso di velocizzare ed ottimizzare gli algoritmi, sebbene abbia richiesto uno sforzo di implementazione molto maggiore. Java e Neo4j sono stati quindi utilizzati per la preparazione dei dati, i quali vengono utilizzati dai programmi in c che eseguono il cuore dell'algoritmo. Il codice C, infine, genera delle matrici o delle istruzioni Matlab per la visualizzazione e comparazione dei risultati.

Alcuni algoritmi, come AsySVD, NNCosNbgr, MovieAVG, PureSVD e TopRated sono stati eseguiti utilizzando delle librerie Matlab sviluppate dal Professor Cremonesi e dal Dott. Roberto Turrin: alcuni di questi hanno un'implementazione Matlab nativa, mentre altri richiamano codice C dal contesto di esecuzione Matlab. Gli algoritmi Learning To Recommend With Social Trust Ensemble[50] e TrustWalker[36] sono stati implementati sia in C che in Java; gli algoritmi CLiMF[79] e ALSPR[81] sono stati implementati in C; ALSPR, che dispone di uno pseudocodice che fa un uso massivo delle matrici, è stato implementato inizialmente Matlab per la sua semplicità; CLiMF invece dispone di uno pseudocodice che si presta particolarmente all'implementazione C.

Capitolo 5

Metodologia e metriche di valutazione

In questo capitolo verranno descritte le metriche utilizzate per valutare e confrontare gli algoritmi nella sezione seguente. Nella seconda sezione verranno descritte le metodologie di testing, ovvero il modo in cui il dataset è stato suddiviso per la fase di apprendimento del modello e per la sua successiva valutazione.

5.1 Metriche di errore

Una metrica di errore è un metodo per calcolare la qualità di un algoritmo di raccomandazione. Le metodologie classiche sono MAE, MSE ed RMSE. Queste sono state molto utilizzate in letteratura per confrontare gli algoritmi, ma non sono in grado di descrivere con precisione gli algoritmi che lavorano con rating impliciti; per confrontare questi algoritmi è necessario introdurre i concetti di recall, fallout e ROC, che sono proprie dell'*information retrieval*. Esse si basano sul principio che non importa quanto l'algoritmo predica con precisione i rating dell'utente, ma quanto questo riesca a produrre una lista di raccomandazione vicina ai gusti dell'utente. Per algoritmi che lavorano con rating impliciti è più sensato utilizzare metriche basate sul ranking come MRR e ARP.

5.1.1 MAE e RMSE

Il *Mean Absolute Error* è una quantità usata in statistica per misurare quanto una predizione si avvicini al valore reale[28]. Nell'ambito dei sistemi di raccomandazione viene utilizzata per misurare quanto il *rating predetto* si discosti dal *valore*

reale. Valori bassi sono indice di prestazioni migliori. Si calcola come:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|}{N} \tag{5.1}$$

Il *Root Mean Square Error* è una misura d'errore frequentemente utilizzata per misurare la differenza tra i valori predetti da un modello e i valori realmente osservati. Nell'ambito dei sistemi di raccomandazione viene utilizzata per misurare la precisione della predizione dei rating.

Si calcola come [14]:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{i,j} - \hat{r}_{i,j})^2}{N}} \tag{5.2}$$

L'*RMSE* è stato utilizzato per valutare le performance degli algoritmi presentati alla competizione proposta da *Netflix* [8].

Recentemente è stato fatto notare che utilizzare il *MAE* ed l'*RMSE* per la valutazione delle prestazioni di un sistema di raccomandazione potrebbe non essere corretto [14][17]. Per questo motivo si è scelto di utilizzare metriche più adeguate per confrontare gli algoritmi esaminati.

5.1.2 Recall

La *Recall* è una classificazione statistica frequentemente utilizzata in diversi ambiti del sapere. In statistica è definita come il numero di veri positivi diviso il numero totale di elementi che attualmente appartengono alla classe. Nell'*information retrieval* la *recall* è definita come il numero di item attinenti recuperati da una ricerca diviso il numero totale di item esistenti. Un valore di recall pari a 1.0 indica che tutti gli item attinenti sono stati recuperati dalla ricerca.

Nell'ambio dei sistemi di raccomandazione la *recall* è calcolata in funzione del numero di item mostrati all'utente (n). Si procede calcolando le predizioni dei rating per tutti gli item non valutati da ogni utente, quindi si selezionano i primi n item ordinati secondo il rating predetto (*top-n item*), infine se l'item compare nella lista delle raccomandazioni si ha una *hit*. Quindi si calcola come[17]:

$$recall(n) = \frac{\#hits(T^+)}{|T^+|} \tag{5.3}$$

Dove T^+ è il set degli item rilevanti. Generalmente in un sistema di raccomandazione il set degli item rilevanti è composto da item a cui è stato assegnato un rating positivo, e viceversa si considerano non rilevanti gli item con un rating

negativo. Si tenga presente che nei dataset presi in esame i rating sono assegnati in una scala compresa nell'intervallo $[1, 5]$, quindi si considerano rilevanti gli item con un rating superiore a 4, e irrilevanti gli item con rating inferiore a 2. Nel resto del lavoro sarà utilizzata la notazione *recall@n* per indicare il valore di recall *recall(n)*.

5.1.3 Fallout

L'interpretazione della *recall* può di per sè essere fuorviante poiché, sebbene rappresenti in maniera efficace il numero di predizioni corrette, non da alcuna informazione sui *falsi positivi*. Per questo motivo si accompagna allo studio della *recall* quello del *fallout*, così da dare una misura di quanti item irrilevanti siano stati erroneamente predetti. Il calcolo è analogo a quello della *recall*, differisce esclusivamente nell'insieme utilizzato come test. Se per il calcolo della recall si utilizza l'insieme dei rating positivi T^+, per il *fallout* si utilizza l'insieme dei rating negativi T^-. Si procede calcolando le predizioni dei rating per tutti gli item non valutati da ogni utente, quindi si selezionano i primi n item ordinati secondo il rating predetto (*top-n item*), infine se l'item compare nella lista delle raccomandazioni si ha una *hit*. Quindi si calcola come[17]:

$$fallout(n) = \frac{\#hits(T^-)}{|T^-|} \tag{5.4}$$

Dove T^- è il set degli item non rilevanti.

5.1.4 ROC

L'analisi di *recall* e *fallout* è facilitata dalle curve *ROC* (*Receiver Operating Characteristic* o anche note come *Relative Operating Characteristic*[10]). Si tratta di schemi grafici per un classificatore binario. Lungo i due assi sono rappresentate la *recall* e il *fallout* (o in un contesto più generale rispettivamente *veri positivi* e *falsi positivi*). La curva di ROC per un sistema di raccomandazione casuale è una retta diagonale che unisce gli estremi del grafico. Per confrontare le curve di ROC viene spesso utilizzata l'area sotto la curva: la bontà di un algoritmo di raccomandazione è proporzionale all'area sotto la sua curva di ROC; un algoritmo di raccomandazione casuale ha un'area sotto la curva di 0.5, che costruisce il limite minimo per la qualità di RS. Valori minori di 0.5 indicano che è preferibile un RS casuale e, quindi, che l'algoritmo non è molto efficace.

5.1.5 Mean Reciprocal Rank (MRR)

Il *Mean reciprocal rank* è una statistica utilizzata per valutare un qualunque processo che produce una lista di possibili risposte ad una *query*, ordinate secondo la probabilità di correttezza. Il *reciprocal rank* di una risposta ad una query è l'inverso del rank della prima risposta corretta. Il *mean reciprocal rank* è la media dei reciproci dei *rank* dei risultati di una generica query Q. Si calcola come:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{5.5}$$

Assume valori compresi nell'intervallo $[0, 1]$, si tende a preferire valori vicini ad 1.

5.1.6 Average Relative Position (ARP)

L'*Average Relative Position* è una statistica utilizzata per valutare un qualunque processo che produce una lista di possibili risposte ad una *query*. Si tratta della media delle posizioni assunte dagli item predetti in una lista ordinata. Valori bassi indicano risposte migliori. Si calcola come:

$$ARP = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_i \tag{5.6}$$

5.2 Metodologie di testing

Nella valutazione di un RS è necessario suddividere i dati a disposizione in modo tale che una parte di essi sia destinata all'apprendimento del modello, *rating set*, e la restante parte per calcolare la metrica di errore, *test set*; il modo in cui i dati vengono suddivisi è chiamato *metodologia di testing*. Esistono vari modi per dividere un dataset; tra i più importanti citiamo il *k-fold cross validation* di cui il *leave one out* è un caso particolare e l'*hold-out*. Il *k-fold* suddivide il dataset in k dataset più piccoli. L'apprendimento poi viene effettuato a rotazione su k-1 di questi dataset, mentre il test sul dataset rimanente. Il termine cross validation si riferisce a come questi risultati vengano validati: effettuando k apprendimenti (e quindi k testing) avremo k valori diversi per le metriche di errore: essi poi verranno mediati oppure saranno scartati quelli meno promettenti oppure verrà preso in considerazione soltanto il risultato migliore. Questa tecnica, sebbene

utile, è molto pesante dal punto di vista computazione, dovendo effettuare k apprendimenti e testing su un dataset grande $frack - 1k$ di quello originale.

5.2.1 Hold-out

L'*hold-out* è un metodo che permette di creare facilmente rating set e test set a partire da un'unica URM. Può funzionare con utenti, item e rating; nel nostro caso si è scelta la suddivisione in base ai rating poiché è il metodo più semplice per il successivo calcolo di recall e fallout. Il metodo, quindi, prevede di estrarre una certa percentuale di rating ed escluderla dal rating set; i rating esclusi poi andranno a formare il test set. In questo modo il numero di utenti e di item viene preservato e il modello viene appreso per ogni utente, così poi da cercare di predire gli altri rating per lui rilevanti (presenti nel test set). Può essere visto come un k-fold in cui viene effettuato soltanto un apprendimento.

Capitolo 6

Social Modeling

Come precedentemente illustrato nell'analisi di *Learning to Recommend*, le informazioni sociali, ovvero il contributo di ogni amico dell'utente target, sono integrate attraverso la funzione di predizione (6.1). Il ruolo teorico del parametro sociale α è quello di indicare quanto l'utente target confidi nei propri gusti piuttosto che nei gusti dei propri amici. Intuitivamente si è portati a pensare che si tratta di un fattore molto importante all'interno di un processo di *decision making*. Questo capitolo è incentrato sullo studio del parametro sociale α e degli effetti delle sue variazioni. Verrà inizialmente presentata un'estensione dell'algoritmo *Social Trust Ensemble* in cui il parametro sociale viene calcolato per ogni utente durante la fase di apprendimento. In seguito si mostrerà uno studio con tecniche di regressione finalizzato a trovare una relazione tra il parametro sociale e il profilo dell'utente. Prendendo spunto dal metodo di integrazione dei dati sociali suggerito da *Learning to Recommend*[50], si è pensato di utilizzare la seguente *funzione di predizione dei rating* all'interno degli algoritmi *CLiMF* [79] e *ALSPR* [81], basati su rating impliciti (descritti in precedenza nelle Sezioni 3.3 e 3.4):

$$\hat{r}_{ui} = f\left\langle U_i, V_j \right\rangle = \alpha(U_i \cdot V_j) + (1-\alpha) \sum_{k \in T_i} \frac{U_k \cdot V_j}{|T_i|} \tag{6.1}$$

6.1 Social trust Ensemble con α personalizzato

La (6.1) proposta dagli autori di *Learning to Recommend* fa un uso statico del parametro sociale, utilizza cioè lo stesso valore per tutti gli utenti del dataset. Si è pensato tuttavia che questo valore sia altamente soggettivo e che vari quindi da utente ad utente, per cui si è scelto di esplorare nuove tecniche per stabilire quale sia l'impatto di un α personalizzato per ogni utente.

Proponiamo una variante dell'algoritmo in cui il parametro α viene appreso durante la fase di training. La funzione obiettivo (3.2) viene minimizzata con un algoritmo di *discesa del gradiente* secondo le (3.4) (3.5). Si applica la discesa del gradiente anche sul parametro α utilizzando la seguente equazione:

$$\alpha_i^{t+1} = \alpha_i^t - \gamma \frac{\partial L}{\partial \alpha} \tag{6.2}$$

É necessario quindi calcolare la derivata rispetto ad α della funzione obiettivo (3.2):

$$\begin{aligned} \frac{\partial L}{\partial \alpha} =& \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R (R_{ij} - g(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j)) \\ & (-g'(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j)) \\ & (U_i^T V_j - \sum_{k \in T(i)} S_{ik} U_k^T V_j) \end{aligned} \tag{6.3}$$

Le derivate di U_i e V_j restano pressocché invariate, l'unica differenza consiste nel fatto che α è si costante, ma diverso per ogni utente, per cui si riscrivono le equazioni:

$$\begin{aligned} \frac{\partial L}{\partial U_i} =& \alpha_i \sum_{j=1}^{n} I_{ij}^R g'(\alpha_i U_i^T V_j + (1-\alpha_i) \sum_{k \in T(i)} S_{ik} U_k^T V_j) V_j \\ & (g(\alpha_i U_i^T V_j + (1-\alpha_i) \sum_{k \in T(i)} S_{ik} U_k^T V_j) - R_{ij}) + \\ & + (1-\alpha_i) \sum_{p \in B(i)} \sum_{j=1}^{n} I_{ij}^R g'(\alpha_p U_p^T V_j + (1-\alpha_p) \sum_{k \in T(p)} S_{pk} U_k^T V_j) V_j \\ & (g(\alpha_p U_p^T V_j + (1-\alpha_p) \sum_{k \in T(p)} S_{pk} U_k^T V_j) - R_{pj}) S_{pi} V_j + \lambda_U U_i \end{aligned} \tag{6.4}$$

$$\begin{aligned} \frac{\partial L}{\partial V_j} =& \sum_{i=1}^{m} I_{ij}^R g'(\alpha_i U_i^T V_j + (1-\alpha_i) \sum_{k \in T(i)} S_{ik} U_k^T V_j) V_j \\ & (g(\alpha_i U_i^T V_j + (1-\alpha_i) \sum_{k \in T(i)} S_{ik} U_k^T V_j) - R_{ij}) \\ & (\alpha_i U_i^T + (1-\alpha_i) \sum_{k \in T(i)} S_{ik} U_k^T) + \lambda_V V_j \end{aligned} \tag{6.5}$$

6.1.1 Analisi della correlazione utente-α

Il calcolo del coefficiente sociale personalizzato appesantisce la fase di training. Si è ritenuto opportuno effettuare uno studio sulla natura dei profili utente per stabilire se esiste una correlazione tra questi e il parametro α ottimo per ogni utente. É stata condotta la seguente analisi sui tre dataset a disposizione:

- è stato utilizzato l'algoritmo *Social Trust Ensemble* nella sua versione originale con diversi valori di α, precisamente con valori in $[0, 1]$ con un passo di 0.1;
- si è ricavato quindi , per ogni coppia *user-item*, il valore di α per il quale il punteggio di recall è maggiore;
- infine sono state applicate tecniche di regressione per desumere possibili funzioni che mettano in relazione il parametro α col numero di amici e di rating di ogni utente, ottenere quindi una funzione $f(\#f, \#r)$ a valori in $[0, 1]$

É stata calcolata sia la regressione lineare sia la regressione quadratica le cui funzioni sono le seguenti e i cui errori possono essere riassunti nella tabella 6.1.

$$\alpha = -1.7305e-04 * \#\mathrm{f} + 2.9914e-03 * \#r + 4.9527e-01 \tag{6.6}$$

$$\alpha = -1.3104e-03 * \#f + 9.1638e-03 * \#r + 5.5364e-01 \tag{6.7}$$

$$\alpha = -2.7547e-03 * \#f + -9.0920e-04 * \#r + 4.8391e-01 \tag{6.8}$$

$$\begin{aligned}\alpha =& 7.7151e-06 * \#f^2 + 6.6624e-05 * \#f * \#r + 2.5876e-04 * \#r^2 + \\ & - 1.0964e-03 * \#f - 1.4160e-03 * \#r + 5.1521e-01\end{aligned} \tag{6.9}$$

$$\begin{aligned}\alpha =& 5.0940e-06 * \#f^2 + 8.7241e-04 * \#f * \#r + 1.4359e-03 * \#r^2 + \\ & - 8.3958e-03 * \#f - 2.1271e-02 * \#r + 6.9473e-01\end{aligned} \tag{6.10}$$

$$\begin{aligned}\alpha =& 1.5427e-04 * \#f^2 + 6.3881e-05 * \#f * \#r + 2.2730e-05 * \#r^2 + \\ & - 7.7436e-03 * \#f - 1.4923e-03 * \#r + 4.9361e-01\end{aligned} \tag{6.11}$$

funzione	last.fm	del.icio.us	Epinions
lineare	0.316	0.316	0.314
quadratica	0.316	0.314	0.314

Tabella 6.1: RMSE calcolato sulle funzioni di regressione

Questo studio ha mostrato l'impossibilità di trovare una funzione che calcoli il miglior α in base al numero di amici e di rating. Le correlazioni lineare e quadratica, infatti, mostrano come tutti i punti si distribuiscano quasi uniformemente su α: ciò conferma l'uniformità dei valori di α e, quindi, l'assenza di correlazione.

6.2 CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering

Per integrare il contributo delle informazioni sociali nell'algoritmo *CLiMF*, è stata sostituita la funzione di predizione proposta dagli autori (3.11) con la funzione utilizzata dagli autori di *Learning to Recommend* (6.1):

$$\begin{aligned} F(U,V) = \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}[\ln g(f(U_i,V_j))+ \\ +\sum_{k=1}^{N}\ln(1-Y_{ij}g(f(U_i,V_k)-f(U_i,V_j)))]+ \\ -\frac{\lambda}{2}(||U||^2+||V||^2) \end{aligned} \tag{6.12}$$

La funzione $f(U_i, V_j)$ è stata precedentemente definita dall'equazione (6.1). Naturalmente è stato necessario ricalcolare le derivate parziali rispetto ad U_i (3.13) e V_j (3.14) con la nuova funzione di predizione, ottenendo quindi:

$$\begin{aligned} \frac{\partial F}{\partial U_i} = &\sum_{j=1}^{N} Y_{ij}[g(-f_{ij})V_j+ \\ &+\sum_{k=1}^{N}\frac{Y_{ik}g'(f_{ik}-f_{ij})}{1-Y_{ik}g(f_{ik}-f_{ij})}(V_j-V_k)]\alpha-\lambda U_i \end{aligned} \tag{6.13}$$

$$\begin{aligned}\frac{\partial F}{\partial V_j} =& Y_{ij}[g(-f_{ij})+ \\ &+\sum_{k=1}^{N} Y_{ik} g'(f_{ij}-f_{ik})(\frac{1}{1-Y_{ik}g(f_{ik}-f_{ij})}+ \\ &-\frac{1}{1-Y_{ij}g(f_{ij}-f_{ik})})][\alpha U_i + (1-\alpha)\sum_{f\in T_i}\frac{U_k \cdot V_j}{|T_i|}] - \lambda V_j\end{aligned} \tag{6.14}$$

Ottimizzazione delle equazioni Durante l'implementazione della (3.14) è emerso che è possibile semplificare il fattore:

$$\frac{1}{1-Y_{ik}g(f_{ik}-f_{ij})} - \frac{1}{1-Y_{ij}g(f_{ij}-f_{ik})} \tag{6.15}$$

É necessario tenere presente che la *funzione logistica* gode della seguente proprietà:

$$g(-x) = \frac{g'(x)}{g(x)} \tag{6.16}$$

Inoltre si noti che il fattore (6.15) contribuisce solo se $Y_{ij} = 1$ e $Y_{ik} = 1$, quindi ne segue che:

$$\begin{aligned}&\frac{1}{1-Y_{ik}g(f_{ik}-f_{ij})} - \frac{1}{1-Y_{ij}g(f_{ij}-f_{ik})} = \\ &= \frac{1}{1-g(f_{ik}-f_{ij})} - \frac{1}{1-g(f_{ij}-f_{ik})} = \\ &= \frac{g(f_{ik}-f_{ij}) - g(f_{ij}-f_{ik})}{1-g(f_{ij}-f_{ik})-g(f_{ik}-f_{ij})+g'(f_{ij}-f_{ik})}\end{aligned} \tag{6.17}$$

Sostituendo l'equazione della funzione logistica (3.3) e della sua derivata, è possibile, dopo una serie di passaggi aritmetici, riscrivere il fattore (6.15) come:

$$1 - \frac{2\ e^{(f_{ik}-f_{ij})}}{e^{(f_{ik}-f_{ij})}+1} \tag{6.18}$$

Il calcolo è così notevolmente semplificato, inoltre si risolve un problema di stabilità numerica dovuto ad una forma indeterminata presente nella forma (6.15).

6.3 ALSPR: Alternating Least Squares for Personalized Ranking

Per l'algoritmo *Alternating Least Squares for Personalized Ranking* [81], descritto nella Sezione 3.4 sono state integrate le informazioni sociali modificando la (3.15) utilizzando come formula per la predizione del rating la (6.19), adattando la (6.1).

$$\hat{r}_{ui} = \alpha \cdot p_u \cdot q_i^T + (1 - \alpha) \frac{1}{|TU_u|} \sum_{f \in TU_u} p_f \cdot q_i \tag{6.19}$$

Nelle equazioni abbrevieremo $\frac{1}{|TU_u|} \sum_{f \in TU_u} p_f$ con $\overline{p}_f$.

La derivata di (3.15) rispetto a p_u quindi è calcolata nella (6.20).

$$
\begin{aligned}
&\frac{\partial f_R(P,Q)}{\partial p_u} = \\
&= \sum_{i\in I} c_{ui} \sum_{j\in I} s_j \left[\alpha p_u q_i^T - (1-\alpha)\overline{p}_f q_i^T - \alpha p_u q_j^T - (1-\alpha)\overline{p}_f q_j^T - (r_{ui} - r_u j)\right] \cdot \alpha \cdot (q_i - q_j) = \\
&= \sum_{i\in I} c_{ui} \sum_{j\in I} s_j \left[\alpha p_u (q_i - q_j)^T + (1-\alpha)\overline{p}_f (q_i - q_j)^T - (r_{ui} - r_u j)\right] \cdot \alpha \cdot (q_i - q_j) = \\
&= \underbrace{\left(\sum_{j\in I} s_j\right)}_{\tilde{1}} \underbrace{\left(\sum_{i\in I} c_{ui} q_i q_i^T\right)}_{\overline{A}} [\alpha^2 p_u + \alpha(1-\alpha)\overline{p}_f] + \\
&\quad - \underbrace{\left(\sum_{i\in I} c_{ui} q_i\right)}_{\overline{q}} \underbrace{\left(\sum_{j\in I} s_j q_j^T\right)}_{\tilde{q}^T} [\alpha^2 p_u + \alpha(1-\alpha)\overline{p}_f] + \\
&\quad - \underbrace{\left(\sum_{j\in I} s_j q_j\right)}_{\tilde{q}} \underbrace{\left(\sum_{i\in I} c_{ui} q_i^T\right)}_{\overline{q}^T} [\alpha^2 p_u + \alpha(1-\alpha)\overline{p}_f] + \\
&\quad + \underbrace{\left(\sum_{i\in I} c_{ui}\right)}_{\overline{1}} \underbrace{\left(\sum_{j\in I} s_j q_j q_j^T\right)}_{\tilde{A}} [\alpha^2 p_u + \alpha(1-\alpha)\overline{p}_f] + \\
&\quad - \underbrace{\left(\sum_{i\in I} c_{ui} q_i r_{ui}\right)}_{\overline{b}} \underbrace{\left(\sum_{j\in I} s_j\right)}_{\tilde{1}} + \underbrace{\left(\sum_{i\in I} c_{ui} q_i\right)}_{\overline{q}} \underbrace{\left(\sum_{j\in I} s_j r_{uj}\right)}_{\tilde{r}} + \\
&\quad + \underbrace{\left(\sum_{i\in I} c_{ui} r_{ui}\right)}_{\overline{r}} \underbrace{\left(\sum_{j\in I} s_j q_j\right)}_{\tilde{q}} - \underbrace{\left(\sum_{i\in I} c_{ui}\right)}_{\overline{1}} \underbrace{\left(\sum_{j\in I} s_j q_j r_{uj}\right)}_{\tilde{b}} = \\
&= \left(\tilde{1}\overline{A} - \overline{q}\tilde{q}^T - \tilde{q}\overline{q}^T + \overline{1}\tilde{A}\right)[\alpha^2 p_u + \alpha(1-\alpha)\overline{p}_f] - \left(\overline{b}\tilde{1} - \overline{q}\tilde{r} - \overline{r}\tilde{q} + \overline{1}\tilde{b}\right)
\end{aligned}
\tag{6.20}
$$

chiamando con M_p l'espressione $\left(\tilde{1}\overline{A} - \overline{q}\tilde{q}^T - \tilde{q}\overline{q}^T + \overline{1}\tilde{A}\right)$ e con y_p l'espressione $\left(\overline{b}\tilde{1} - \overline{q}\tilde{r} - \overline{r}\tilde{q} + \overline{1}\tilde{b}\right)$ ed egualgiando la derivata a 0 otteniamo la (6.21).

$$
p_u = \frac{M_p^{-1} y_p - (1-\alpha)\overline{p}_f}{\alpha} \tag{6.21}
$$

L'equazione di aggiornamento della matrice P, quindi, viene modificata tenendo conto di alfa e di $\overline{p}_f$. Per l'implementazione si è deciso di creare una matrice $\overline{P}_f$

che memorizza per ogni utente u la media dei propri amici. Alla prima epoca essa non viene considerata e l'equazione utilizzata per l'aggiornamento di P è quella originale. Dopo questo aggiornamento viene calcolata la matrice $\overline{P}_f$ che varrà utilizzata nel successivo aggiornamento di Q e nella prossima epoca per P. Ad ogni epoca questa matrice viene aggiornata, dopo aver calcolato P. Per quanto riguarda la derivata rispetto a q_i otteniamo la (6.22).

$$\begin{aligned}
&\frac{\partial f_R(P,Q)}{\partial q_i} = \\
&= \sum_{u \in U} \sum_{j \in I} c_{ui} s_j [\alpha p_u + (1-\alpha)\overline{p}_f] \left\{ [\alpha p_u + (1-\alpha)\overline{p}_f]^T (q_i - q_j) - (r_{ui} - r_{uj}) \right\} + \\
&\quad + \sum_{u \in U} \sum_{j \in I} c_{ui} s_j \left\{ -[\alpha p_u + (1-\alpha)\overline{p}_f] \right\} \left\{ [\alpha p_u + (1-\alpha)\overline{p}_f]^T (q_j - q_i) - (r_{uj} - r_{ui}) \right\}
\end{aligned} \tag{6.22}$$

Sviluppando la precedente equazione si arriva ad un'espressione sostanzialmente

simile alla (3.17), dove però valgono le definizioni (6.23) e (6.24).

$$
\begin{aligned}
\overline{A} &= \alpha^2 \underbrace{\left(\sum_{u \in U} c_{ui} p_u p_u^T \right)}_{\overline{A}_{uu}} + \alpha(1-\alpha) \left(\underbrace{\sum_{u \in U} c_{ui} p_u \overline{p}_f^T}_{\overline{A}_{uf}} + \underbrace{\sum_{u \in U} c_{ui} \overline{p}_f p_u^T}_{\overline{A}_{fu}} \right) + (1-\alpha)^2 \underbrace{\left(\sum_{u \in U} c_{ui} \overline{p}_f \overline{p}_f^T \right)}_{\overline{A}_{ff}} \\
\overline{b} &= \alpha \underbrace{\left(\sum_{u \in U} c_{ui} p_u r_{ui} \right)}_{\overline{b}_u} + (1-\alpha) \underbrace{\left(\sum_{u \in U} c_{ui} \overline{p}_f r_{ui} \right)}_{\overline{b}_f} \\
\overline{\overline{p}}_1 &= \alpha \underbrace{\left(\sum_{u \in U} c_{ui} \overbrace{\left(\sum_{j \in I} s_j r_{uj} \right)}^{\tilde{r}} p_u \right)}_{\overline{\overline{p}}_{1u}} + (1-\alpha) \underbrace{\left(\sum_{u \in U} c_{ui} \overbrace{\left(\sum_{j \in I} s_j r_{uj} \right)}^{\tilde{r}} \overline{p}_f \right)}_{\overline{\overline{p}}_{1f}} \\
\overline{\overline{p}}_2 &= \alpha^2 \underbrace{\left(\sum_{u \in U} p_u p_u^T \overbrace{\left(\sum_{j \in I} c_{uj} q_j \right)}^{\overline{q}} \right)}_{\overline{\overline{p}}_{2uu}} + \alpha(1-\alpha) \left(\underbrace{\sum_{u \in U} p_u \overline{p}_f^T \overbrace{\left(\sum_{j \in I} c_{uj} q_j \right)}^{\overline{q}}}_{\overline{\overline{p}}_{2uf}} + \underbrace{\sum_{u \in U} \overline{p}_f p_u^T \overbrace{\left(\sum_{j \in I} c_{uj} q_j \right)}^{\overline{q}}}_{\overline{\overline{p}}_{2fu}} \right) + \\
&+ (1-\alpha)^2 \underbrace{\left(\sum_{u \in U} \overline{p}_f \overline{p}_f^T \overbrace{\left(\sum_{j \in I} c_{uj} q_j \right)}^{\overline{q}} \right)}_{\overline{\overline{p}}_{2ff}} \\
\overline{\overline{p}}_3 &= \alpha \underbrace{\left(\sum_{u \in U} \overbrace{\left(\sum_{j \in I} c_{uj} r_{uj} \right)}^{\overline{q}} p_u \right)}_{\overline{\overline{p}}_{3u}} + (1-\alpha) \underbrace{\left(\sum_{u \in U} \overbrace{\left(\sum_{j \in I} c_{uj} r_{uj} \right)}^{\overline{q}} \overline{p}_f \right)}_{\overline{\overline{p}}_{3f}} \\
\overline{\overline{b}} &= \alpha \underbrace{\left(\sum_{u \in U} p_u r_{ui} \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}} \right)}_{\overline{\overline{b}}_u} + (1-\alpha) \underbrace{\left(\sum_{u \in U} \overline{p}_f r_{ui} \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}} \right)}_{\overline{\overline{b}}_f}
\end{aligned}
\tag{6.23}
$$

$$\overline{\overline{A}} = \alpha^2 \underbrace{\left(\sum_{u \in U} p_u p_u^T \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}} \right)}_{\overline{\overline{A}}_{uu}} +$$

$$+ \alpha(1-\alpha) \left(\underbrace{\sum_{u \in U} p_u \overline{p}_f^T \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}}}_{\overline{\overline{A}}_{uf}} + \underbrace{\sum_{u \in U} \overline{p}_f p_u^T \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}}}_{\overline{\overline{A}}_{fu}} \right) + (1-\alpha)^2 \underbrace{\left(\sum_{u \in U} \overline{p}_f \overline{p}_f^T \overbrace{\left(\sum_{j \in I} c_{uj} \right)}^{\overline{1}} \right)}_{\overline{\overline{A}}_{ff}}$$

(6.24)

L'equazione di aggiornamento di Q invece resta invariata. Sono alcuni coefficienti, come ad esempio $\overline{\overline{A}}$ e $\overline{A}$, che vengono modificati per tenere conto delle informazioni sociali.

6.4 Risultati

In questa sezione vengono confrontati i risultati delle simulazioni degli algoritmi *Alspr Social* e *CLiMF Social* con gli algoritmi dello stato dell'arte.

Ognuno dei due algoritmi viene confrontato con la corrispondente versione non sociale, inoltre si effettua un confronto aggiuntivo con gli algoritmi *PureSVD*, *COS* e *TopRated*, questi ultimi sono stati eseguiti sia su una rating matrix binaria sia su rating espliciti, nei grafici si mostra solo la versione dell'algoritmo che ha dato risultati migliori in termini di recall (per esempio si riporterà *PureSVD Esplicito* se avrà una recall maggiore di *PureSVD Implicito*).

Infine viene effettuato un confronto anche con l'algoritmo sociale *LTR with Social Trust Ensemble*.

In Figura 6.1 sono riassunte le fasi di simulazione.

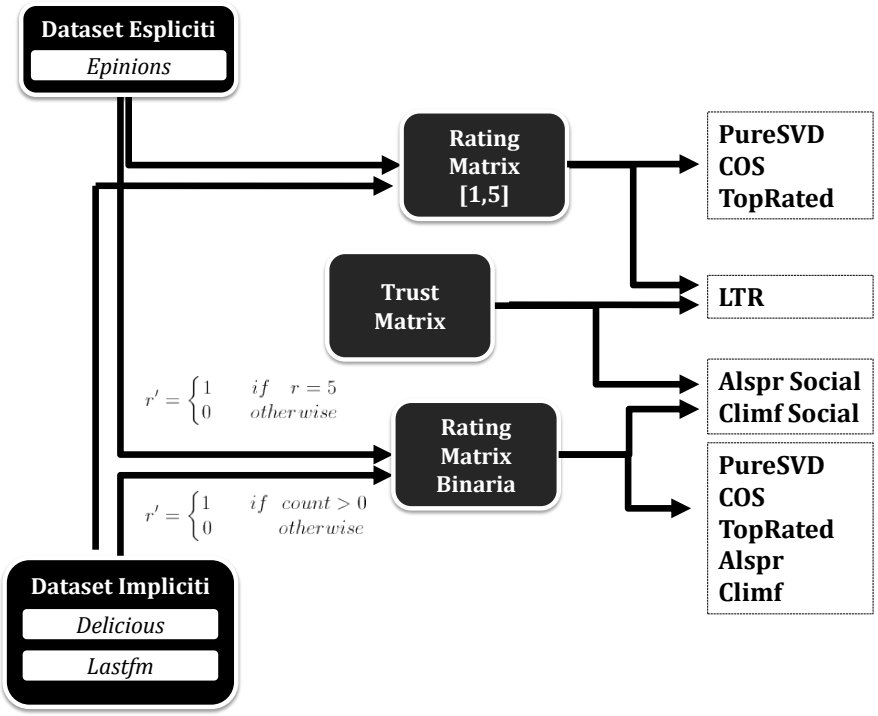


Figura 6.1: Costruzione dei risultati: fasi di simulazione

Questa analisi non prevede il calcolo del fallout poiché, trattandosi di rating impliciti sarebbe concettualmente sbagliato dato che non si dispone di informazioni riguardanti rating negativi. Per ognuno dei tre dataset si mostrano i grafici delle recall.

lastFM Il grafico in Figura 6.2a mostra il calcolo della recall di ogni algoritmo esaminato.

L'asse delle ascisse rappresenta la dimensione della lista di raccomandazione, l'asse delle ordinate il valore di recall della lista di raccomandazione di dimensione n.

Si utilizza un colore diverso per ogni algoritmo, il tratto continuo si riferisce all'algoritmo nella versione non sociale, mentre il tratteggiato alla variante sociale.

Per le varianti sociali è specificato il valore del parametro α usato per la simulazione. Ad esempio la didascalia *Alspr NSB Social 0.9* si riferisce all'algoritmo *Alspr* nella variante *Non support Based*, in versione sociale e con $\alpha = 0.9$, essendo sociale si utilizza un tratto discontinuo.

Il confronto con gli algoritmi di riferimento è stato effettuato con le versioni esplicite di *PureSVD*, *COS* e *TopRated* che hanno prodotto risultati migliori in termini di recall rispetto alle versioni implicite (rispettivamente un incremento di 0.35, 0.31 e 0.15 recall@10), e con *Alspr Non Support Based* che ha prodotto una recall più alta rispetto alla variante *Support Weighted* (+0.21 recall@10).

Osservando il Grafico 6.2a che segue si nota che l'algoritmo CLiMF ha beneficiato della modifica sociale (+0.044 recall@10), tuttavia confrontandolo con gli altri algoritmi si tratta di quello che ha avuto risultati peggiori.

Questi risultati in termini di recall sono confermati dai valori di APR e MRR (Tabelle 6.2, 6.4 e 6.3): si nota un miglioramento di 0.027 di Climf Social rispetto alla versione base, per Alspr solamente 0.016.

L'algoritmo che ha prodotto risultati migliori è Alspr, superando anche gli algoritmi di riferimento sopra citati, tuttavia la componente sociale non ha migliorato significativamente la qualità della raccomandazione.

Delicious Il confronto con gli algoritmi di riferimento è stato effettuato con le versioni esplicite di *PureSVD*, *COS* e *TopRated* che hanno prodotto risultati migliori in termini di recall rispetto alle versioni implicite (rispettivamente un incremento di 0.4, 0.34 e 0.22 recall@10), e con *Alspr Support Weighted* che ha prodotto una recall più alta rispetto alla variante *Non Support Based* (+0.012 recall@10).

Osservando il Grafico 6.2b si nota che sia l'algoritmo CLiMF sia l'algoritmo Alspr hanno beneficiato della modifica sociale (rispettivamente +0.03 e +0.05 recall@10). Tuttavia rispetto agli algoritmi di riferimento i risultati sono insoddisfacenti: ad esempio *CLiMF Social* ha una recall inferiore di 0.4 rispetto a *PureSVD*.

Climf nella versione sociale ha un MRR migliore di 0.01 rispetto alla versione di base (Tabelle 6.2), mentre per Alspr Social il miglioramento non è significativo (Tabelle 6.4).

Epinions Il confronto con gli algoritmi di riferimento è stato effettuato con le versioni esplicite di *PureSVD*, *COS* e *TopRated* che hanno prodotto risultati migliori in termini di recall rispetto alle versioni implicite (rispettivamente un incremento di 0.1, 0.13 e 0.03 recall@10), e con *Non Support Based* che ha prodotto una recall più alta rispetto alla variante *Alspr Support Weighted* (+0.03 recall@10).

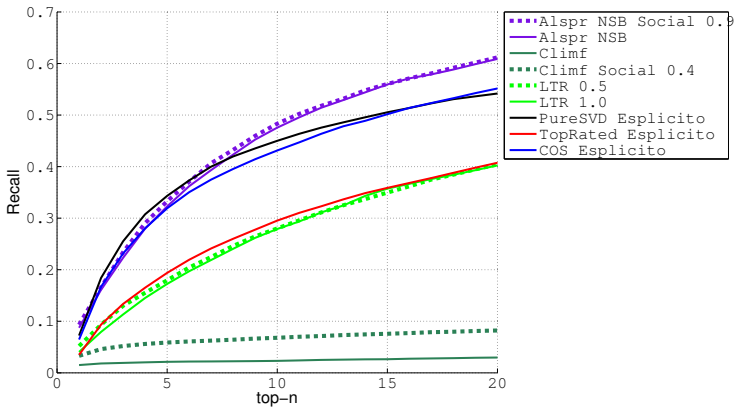
Osservando il Grafico 6.2c si nota che sia l'algoritmo CLiMF sia l'algoritmo Alspr non hanno beneficiato della modifica sociale, in entrambi i casi la variazione di recall è pressocché irrilevante, risultato confermato dai valori di APR e MRR (Tabelle 6.2, 6.4 e 6.3); inoltre entrambi hanno prodotto risultati peggiori rispetto

agli algoritmi di riferimento: ad esempio *Alspr Social* ha una recall inferiore di 0.07 rispetto a *LTR 0.0*.

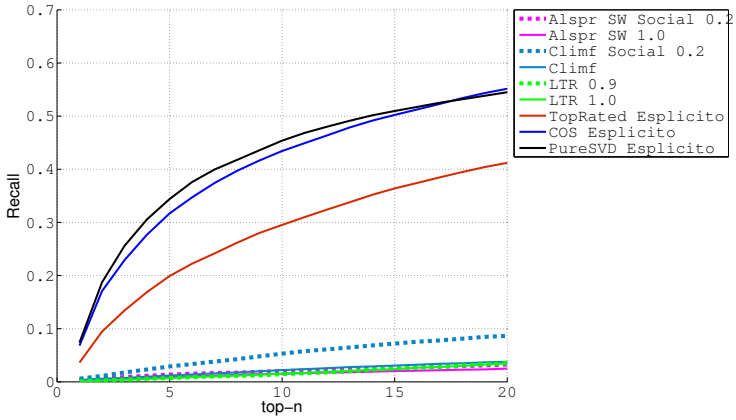
α	last.fm		del.icio.us		Epinions	
	ARP	MRR	ARP	MRR	ARP	MRR
0.1	**464.97**	0.038	500.05	**0.023**	487.29	0.020
0.2	479.74	0.044	500.18	**0.023**	**487.05**	0.020
0.3	479.03	0.043	499.78	**0.023**	488.98	0.020
0.4	493.19	**0.048**	498.97	**0.023**	489.05	0.020
0.5	487.25	0.043	498.23	0.022	489.83	**0.021**
0.6	488.57	0.032	497.09	0.020	491.11	0.020
0.7	505.65	0.024	496.18	0.018	491.30	0.020
0.8	492.67	0.019	495.10	0.016	491.42	0.020
0.9	493.77	0.015	494.51	0.014	492.03	0.020
1.0	492.96	0.021	**494.05**	0.013	490.94	**0.021**

Tabella 6.2: ARP e MRR dell'algoritmo CLiMF calcolate sui dataset a disposizione. $\alpha = 1$ *puramente collaborativo,* $\alpha = 0$ *puramente sociale.*

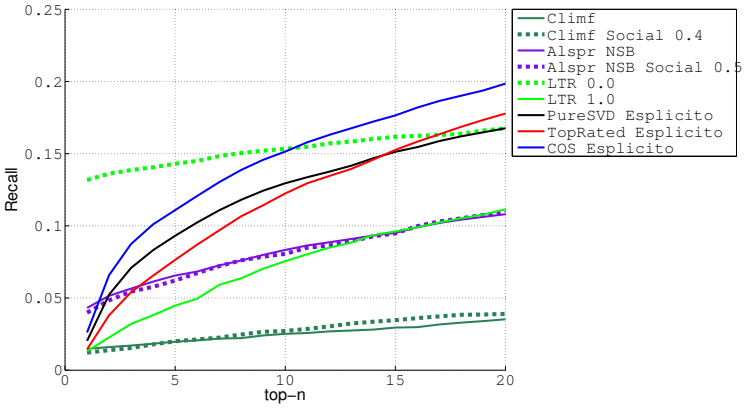
Discussione dei risultati In conclusione, dall'analisi dei risultati si può dedurre come il *social modeling* non migliori significativamente la qualità degli algoritmi esaminati: sia *Climf Social*, sia *Alspr Social* non hanno prodotto risultati significativamente migliori in tutti i dasaset. É interessante notare che nel dataset di Epinions l'algoritmo LTR ha particolarmente beneficiato della componente puramente sociale (rispetto alla versione puramente collaborativa ha un incremento di recall di 0.075): ciò è dovuto al fatto che considerare i profili degli amici per la raccomandazione corrisponde in un certo senso a rendere più densa la matrice dei rating, poichè si include nel profilo dell'utente il *rating behaviour* degli amici. L'esito del *social modeling* ha portato a pensare ad un differente approccio di inclusione delle informazioni sociali in un sistema di raccomandazione, il *social preprocessing*, che sarà discusso nel capitolo seguente.



(a) LastFM



(b) Delicious



(c) Epinions

Figura 6.2: Curve di recall per algoritmi basati su rating impliciti

α	last.fm		del.icio.us		Epinions	
	NSB	SW	NSB	SW	NSB	SW
0.1	121.25	143.73	**497.78**	486.17	427.03	434.19
0.2	114.39	133.02	498.58	478.65	423.58	431.10
0.3	109.21	125.36	499.44	472.34	421.24	**430.30**
0.4	105.49	119.77	500.17	468.01	**420.60**	430.46
0.5	103.10	117.07	500.33	465.74	422.38	430.25
0.6	101.82	**116.51**	500.49	464.50	425.07	431.40
0.7	101.21	117.47	500.58	**464.38**	427.20	433.43
0.8	**101.16**	119.36	500.87	464.54	429.24	435.61
0.9	101.52	121.93	501.20	464.62	430.39	436.85
1.0	102.14	124.98	502.26	464.62	429.73	435.99

Tabella 6.3: ARP dell'algoritmo ALSPR calcolata su diversi dataset, con le varianti popolare (SW) e non popolare (NSB). $\alpha = 1$ *puramente collaborativo,* $\alpha = 0$ *puramente sociale.*

α	last.fm		del.icio.us		Epinions	
	NSB	SW	NSB	SW	NSB	SW
0.1	0.196	0.096	**0.005**	**0.013**	0.056	0.042
0.2	0.206	0.102	**0.005**	0.012	0.057	0.042
0.3	0.212	0.109	0.004	0.012	0.057	0.042
0.4	0.217	0.116	0.004	0.011	0.057	0.042
0.5	0.219	0.119	0.004	0.010	0.057	0.042
0.6	**0.220**	0.121	0.003	0.010	0.057	0.042
0.7	0.219	**0.122**	0.003	0.010	0.056	0.041
0.8	0.215	**0.122**	0.003	0.009	0.055	0.041
0.9	0.210	**0.122**	0.003	0.009	0.055	0.041
1.0	0.203	0.120	0.004	0.010	**0.060**	**0.045**

Tabella 6.4: MRR dell'algoritmo ALSPR calcolata su diversi dataset, con le varianti popolare (SW) e non popolare (NSB). $\alpha = 1$ *puramente collaborativo,* $\alpha = 0$ *puramente sociale.*

Capitolo 7

Social Preprocessing

7.1 Descrizione dell'approccio

Il preprocessing della matrice dei rating si basa sullo stesso principio della modifica degli algoritmi per includere la componente sociale, soltanto che invece di modificare l'*apprendimento del modello*, si modificano i *dati di input* dell'algoritmo. Esso può essere applicato a qualsiasi algoritmo che abbia come input la matrice dei rating. Gli algoritmi descritti di seguito sono pensati per lavorare con rating espliciti, ma il ragionamento può essere esteso anche al caso di rating impliciti. Il parametro α controlla il contributo delle informazioni sociali: se $\alpha = 1$ la matrice resta intatta e quindi l'algoritmo viene eseguito sui dati originali. Se $\alpha = 0$, invece, al posto dei rating dell'utente viene presa in considerazione la media dei rating degli amici. Con α che varia tra 0 ed 1 invece si effettua una *media pesata* tra i rating dell'utente e quelli dei suoi amici, secondo la (7.1).

$$r'_{ui} = \alpha \cdot r_{ui} + (1 - \alpha)\frac{1}{|TU_u|}\sum_{f \in TU} r_{fi} \tag{7.1}$$

Per brevità chiameremo $\frac{1}{|TU_u|}\sum_{f \in TU} r_{fi}$ con $avgF_u$. Vi sono però molti modi di mediare i contributi dell'utente in questione e degli amici e per questo sono stati implementati 4 modi diversi per precalcolare la matrice dei rating descritti nella tabella 7.1 che spiega il comportamento di ciascun algoritmo per ogni coppia utente-item $\langle u, i \rangle$.

algoritmo 1 Questo algoritmo effettua la media pesata soltanto nel caso in cui siano presenti sia il rating dell'utente sia almeno un rating tra quelli degli amici. In questo modo la scala dei rating viene preservata e aumenta la densità della URM.

Caso	alg #1	alg #2	alg #3	alg #4
l'utente corrente non ha rating su i, ma almeno 1 dei suoi amici lo possiede	$avgF_u$	$(1-\alpha)avgF_u$	$\alpha \cdot \bar{\mathbf{r}}_{\mathbf{ui}} + (1-\alpha)avgF_u$	r_{ui}
l'utente corrente possiede un rating su i e almeno 1 dei suoi amici lo possiede	$\alpha \cdot r_{ui} + (1-\alpha)avgF_u$	$\alpha \cdot r_{ui} + (1-\alpha)avgF_u$	$\alpha \cdot r_{ui} + (1-\alpha)avgF_u$	$\alpha \cdot r_{ui} + (1-\alpha)avgF_u$
l'utente corrente possiede un rating su i e nessuno dei suoi amici lo possiede	r_{ui}	r_{ui}	r_{ui}	r_{ui}

Tabella 7.1: Algoritmi di precalcolo della URM

algoritmo 2 Questo algoritmo nel caso in cui almeno un amico abbia espresso una preferenza sull'item in questione effettua comunque la media pesata, sia nel caso in cui l'utente non possieda il rating sia nel caso in cui lo possegga. Nel caso in cui l'utente u non abbia dato un rating all'item i mediare con α comporta un cambio di scala dei rating. Si pensi ad esempio al caso in cui $\alpha = 0.5$: anche nel caso in cui il contributo degli amici produca un rating pari a 5 (quindi un'alta preferenza) la mediazione produrrebbe un rating pari a 2.5, quindi l'algoritmo considera un contributo decisamente positivo come un giudizio lievemente negativo. Anche quest'algoritmo rende più densa la URM.

algoritmo 3 Questo algoritmo effettua sempre la media solo che, a differenza del precedente, invece di mediare con 0 nel caso in cui l'utente u non possegga il rating, effettua la media pesata con la media dei rating di u, denotata con $\bar{\mathbf{r}}_{\mathbf{ui}}$. Questo algoritmo non cambia la scala dei rating e rende più densa la URM.

algoritmo 4 Questo algoritmo, a differenza degli altri, non cambia la densità della URM; esso cambia soltanto i valori dei rating esistenti effettuando una media pesata tra il rating attuale e la media dei rating degli amici dell'utente; quindi né la densità né la scala dei rating vengono cambiate, viene soltanto effettuata una correzione del valore dei rating esistenti.

Ogni algoritmo quindi si differenzia dagli altri soltanto per il modo di gestire il caso in cui l'utente u non possiede rating sull'item i mentre almeno un amico lo possiede. Per capire meglio quante volte si verifica questa situazione è stato effettuato uno studio sui dataset che può essere riassunto nella Tabella 7.2. La tabella è stata costruita contando per ogni rating sull'item i posseduto da amici

Caso	last.fm	delicious	epinions
l'utente corrente non ha rating su i, ma almeno 1 dei suoi amici lo possiede	94.44%	99.57%	99.18%
l'utente corrente possiede un rating su i e almeno 1 dei suoi amici lo possiede	5.56%	0.43%	0.82%

Tabella 7.2: Casistiche dei dataset. Percentuale condizionata dal fatto che almeno 1 amico possegga il rating sull'item considerato

dell'utente u quante volte esso è posseduto anche da u; ad esempio nel caso di last.fm il 99.44% dei casi in cui gli amici dell'utente u posseggono un rating sull'item i, l'utente u non ha espresso un rating su i. Come è evidente dalla tabella, il caso in cui gli algoritmi si differenziano è il più frequente; gli algoritmi, quindi, generano output differenti in modo sostanziale.

7.2 Risultati: Preprocessing della matrice dei rating

In questa sezione vengono discussi i risultati ottenuti dalle simulazioni sui dataset a disposizione in termini di *recall* e *curve di ROC*.

Per ogni dataset è stata eseguita ognuna delle quattro varianti di preprocessing della matrice dei rating espliciti. Le matrici sociali così ottenute sono state utilizzate come input degli algoritmi dello stato dell'arte *PureSVD, COS, TopRated, AsySVD* e *MovieAVG*.

I risultati sono stati confrontati con le corrispondenti versioni non sociali e l'algoritmo sociale *LTR*.

In Figura 7.1 sono riassunte le fasi di simulazione.

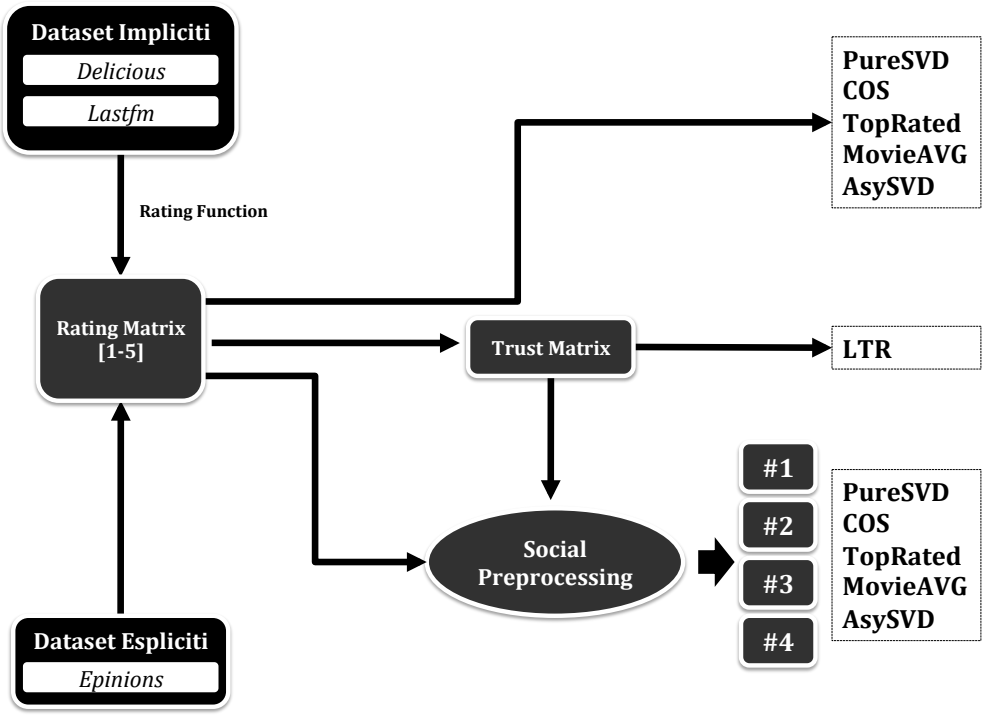


Figura 7.1: Costruzione dei risultati: fasi di simulazione

7.2.1 LastFM

Precalcolo della URM: variante 1 Il grafico in Figura 7.2 mostra il calcolo della recall di ogni algoritmo esaminato. L'asse delle ascisse rappresenta la dimensione della lista di raccomandazione, l'asse delle ordinate il valore di recall della lista di raccomandazione di dimensione n. Il grafico in Figura 7.3 mostra le curve di ROC di ogni algoritmo esaminato. Sull'asse delle ascisse è riportato il fallout, sull'asse delle ordinate la recall. Si utilizza un colore diverso per ogni algoritmo, il tratto continuo si riferisce all'algoritmo nella versione non sociale, mentre il tratteggiato alla variante sociale. Per le varianti sociali è specificato il valore del parametro α usato per la simulazione. Ad esempio la didascalia *PureSVD 0.9* si riferisce all'algoritmo *PureSVD*, in versione sociale con preprocessing e con $\alpha = 0.9$, essendo sociale si utilizza un tratto discontinuo.

Il grafico 7.2 mostra come il preprocessing della matrice dei rating con la **variante 1** incrementi sensibilmente la recall per gli algoritmi *PureSVD* e *TopRated* (rispettivamente un miglioramento di 0.1 e di 0.07 sulla *Recall@10*), si veda la tabella 7.3, tuttavia osservando le curve ROC nella figura 7.3 si nota che in corri-

spondenza dell'aumento della recall si ha un aumento del fallout, ovvero dei falsi positivi, come conferma anche la tabella 7.3. Quindi non si può considerare un miglioramento. Per gli altri algoritmi non si è registrato alcun miglioramento, in alcuni casi il preprocessing ha dato risultati peggiori della versione originale, come per esempio *AsySVD*.

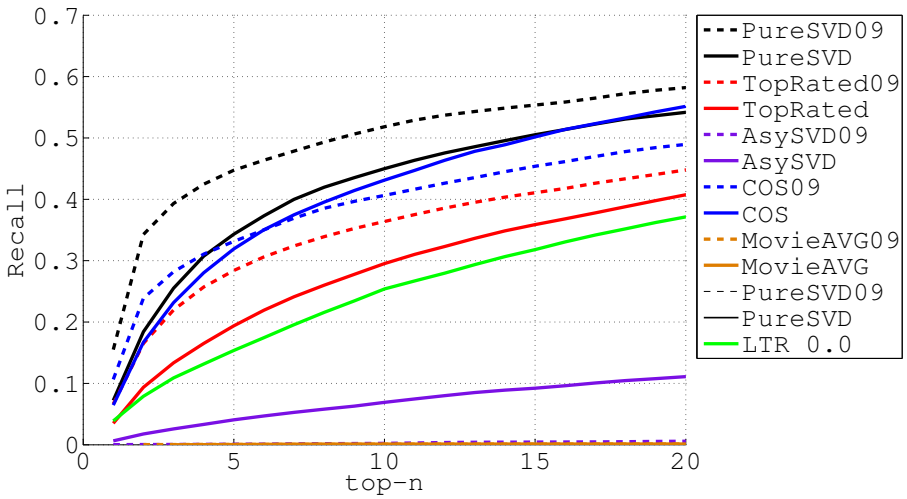


Figura 7.2: LastFM: Confronto Recall algoritmo 1

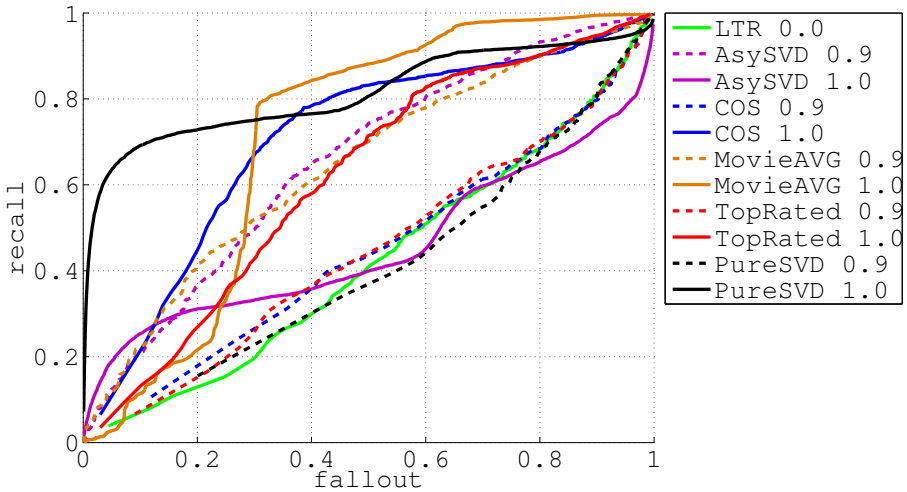


Figura 7.3: LastFM: Confronto curve ROC algoritmo 1

Precalcolo della URM: variante 2 Il grafico mostra come il preprocessing della matrice dei rating con la **variante 2** incrementi sensibilmente la recall per tutti gli algoritmi esaminati, soprattuto l'algoritmo *AsySVD* con un miglioramento di **0.15** (*recall@20*), si veda la tabella 7.3,. Accostando un'analisi delle curve ROC e della tabella 7.3 si nota che l'algoritmo *AsySVD* nella sua versione sociale con preprocessing è più efficace della versione originale. Tuttavia lo stesso miglioramento non si riscontra negli altri algoritmi che hanno un alto numero di falsi positivi. Rispetto ad *LTR with Social Trust Ensemble*, *AsySVD* nella versione sociale è migliore nel raccomandare la top-5 (per dettagli si vedano i grafici B.1c e B.1d in appendice).

Precalcolo della URM: variante 3 Il grafico mostra come il preprocessing della matrice dei rating con la **variante 3** incrementi la recall per gli algoritmi *PureSVD* e *TopRated* (rispettivamente un miglioramento di 0.08 e di 0.07 sulla *Recall@10*), si veda la tabella 7.3, tuttavia osservando le curve ROC si nota che in corrispondenza dell'aumento della recall si ha un aumento del fallout, ovvero dei falsi positivi. Quindi non si può considerare un miglioramento (per dettagli si vedano i grafici B.1e e B.1f in appendice).

Precalcolo della URM: variante 4 Da un'analisi del grafico si evince che il preprocessing della matrice dei rating con la **variante 4** non comporta un incremento della recall. Osservando le curve di ROC si riscontra un peggioramento generale di tutti gli algoritmi, come evidenziato anche nella tabella 7.3 (per dettagli si vedano i grafici B.1g e B.1h in appendice).

Discussione dei risultati Gli algoritmi di preprocessing non hanno prodotto miglioramenti significativi su questo dataset. In generale si è riusciti ad aumentare la recall, ma si è registrato contemporaneamente un peggioramento del fallout. Ciò è attribuibile alla rating function utilizzata (4.1); questa infatti ordina gli artisti ascoltati dall'utente dal più ascoltato (a cui è assegnato un rating di 5) al meno ascoltato, assegnando progressivamente rating più bassi. Per un utente con molti rating vengono assegnati rating negativi ad artisti con numero di ascolti relativamente basso rispetto agli altri dell'utente, ma che in assoluto portebbe avere un numero di ascolti alto (questo vale per la maggior parte dei rating del dataset, si veda 4.2).

La funzione di rating si comporta impropriamente in presenza di utenti che hanno ascoltato un numero elevato di artisti: è probabile che vengano assegnati rating negativi ad artisti sebbene questi abbiano un numero di ascolti elevato, poichè la funzione confronta il numero di ascolti di ogni artista ascoltato dall'utente con il numero massimo di ascolti dell'artista in prima posizione nella

Algoritmo	**variante**	**ARP**	**MRR**	**Recall@10**	**Fallout@10**
AsySVD	base	543.44	0.034	0.072	0.010
	pre 1	574.18	0.004	0.002	0.002
	pre 2	**376.26**	**0.170**	**0.229**	0.268
	pre 3	551.29	0.004	0.001	**0.000**
	pre 4	543.51	0.035	0.073	0.019
COS	base	**138.29**	0.231	0.440	**0.012**
	pre 1	194.95	0.281	0.411	0.447
	pre 2	171.74	**0.322**	**0.470**	0.574
	pre 3	194.37	0.284	0.409	0.460
	pre 4	**138.19**	0.234	0.442	0.618
MovieAVG	base	> 1000	0.002	0.001	0.010
	pre 1	> 1000	0.002	0.001	0.002
	pre 2	> 1000	0.003	0.001	0.000
	pre 3	> 1000	0.003	0.001	0.000
	pre 4	> 1000	0.002	0.001	0.002
PureSVD	base	186.62	0.246	0.457	**0.012**
	pre 1	**151.04**	0.382	0.524	0.673
	pre 2	**151.76**	**0.390**	**0.551**	0.700
	pre 3	149.12	**0.389**	0.526	0.686
	pre 4	185.94	0.247	0.457	0.641
TopRated	base	**199.82**	0.145	0.304	**0.011**
	pre 1	212.24	0.214	**0.370**	0.407
	pre 2	212.16	**0.217**	**0.370**	0.407
	pre 3	212.39	0.215	0.367	0.407
	pre 4	**199.64**	0.147	0.303	0.373
LTR $\alpha = 0.0$	base	212.26	0.108	0.254	0.340

Tabella 7.3: risultati riassuntivi del precalcolo della URM su last.fm

classifica dell'utente, questo comporta che gli artisti in posizioni più basse sono considerati negativamente indipendentemente dal loro numero di ascolti assoluto. Questo scenario si presenta con frequenza considerevole, poiché il 97% degli utenti ha più di 45 rating (Figura 4.3). In questo modo i brani ascoltati dagli amici dell'utente vengono raccomandati, ma contribuiscono ad aumentare il fallout se non sono tra i più ascoltati in assoluto dall'utente.

7.2.2 Delicious

Precalcolo della URM: variante 1 L'analisi della recall mostra come il preprocessing della matrice dei rating con la **variante 1** sia peggiore per tutti gli algoritmi esaminati tranne che per l'algoritmo *MovieAVG* (per dettagli si veda il grafico B.2a in appendice).

Precalcolo della URM: variante 2 Osservando il Grafico 7.4 si può notare come il preprocessing della matrice dei rating con la **variante 2** migliori in termini di recall l'esecuzione degli algoritmi *PureSVD* (+0.45), *COS* (+0.16), *TopRated* (+0.36) e *AsySVD* (+0.22). Osservando le curve ROC (Figura 7.5) e la Tabella 7.4 si nota un effetivo miglioramento per gli algoritmi *PureSVD*, *COS* e *TopRated*; su *AsySVD* invece si registra un alto numero di falsi positivi.

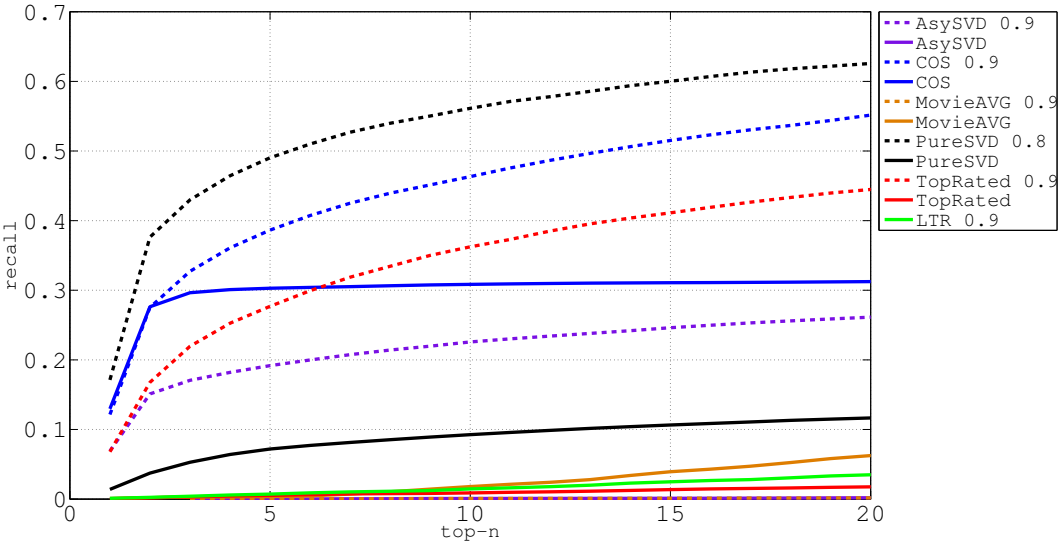


Figura 7.4: Delicious: Confronto Recall algoritmo 2

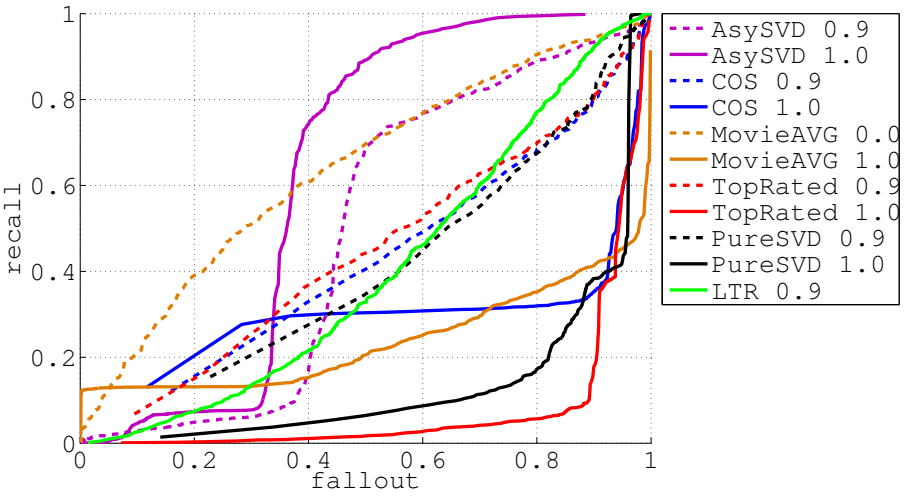


Figura 7.5: Delicious: Confronto curve ROC algoritmo 2

Precalcolo della URM: variante 3 Da un'analisi della recall si può notare come il preprocessing della matrice dei rating con la **variante 3** migliori in termini di recall l'esecuzione degli algoritmi *PureSVD* (+0.45), *COS* (+0.1) e *TopRated* (+0.36). Tale miglioramento è confermato dall'analisi delle curve di ROC. La variante 3 non ha prodotto alcuna variazione rilevante per *MovieAVG* e *AsySVD* (per dettagli si vedano i grafici B.2e e B.2f in appendice).

Precalcolo della URM: variante 4 Da un'analisi della recall si può notare come il preprocessing della matrice dei rating con la **variante 4** migliori in termini di recall l'esecuzione degli algoritmi *PureSVD* (+0.35), *COS* (+0.14) e *TopRated* (+0.3). Tale miglioramento è confermato dall'analisi delle curve di ROC. La variante 4 non ha prodotto alcuna variazione rilevante per *MovieAVG* e *AsySVD* (per dettagli si vedano i grafici B.2g e B.2h in appendice).

Discussione dei risultati Eccetto che per la variante 1, che ha prodotto valori di recall inferiori alle versioni originali (circa -0.1 recall@10), il social preprocessing ha dato risultati molto promettenti su questo dataset (ad esempio la variante 2 migliora PureSVD di +0.45 recall@10).

La variante 2 ha contribuito ad aumentare la recall perché tende ad aumentare la densità della matrice con valori negativi dove l'utente non ha espresso alcuna preferenza; esso quindi effettua, di fatto, un prefiltraggio dei rating, assegnando a rating non noti valori in generale bassi, consolidando, invece, rating alti effettivamente assegnati dall'utente: in questo modo gli algoritmi vengono "tarati" su

Algoritmo	variante	ARP	MRR	Recall@10	Fallout@10
AsySVD	base	747.68	0.002	0.001	0.019
	pre 1	769.19	0.003	0.003	**0.002**
	pre 2	**376.23**	**0.170**	**0.229**	0.270
	pre 3	568.60	0.004	0.003	**0.002**
	pre 4	538.67	0.039	0.078	0.021
COS	base	421.70	0.282	0.309	0.618
	pre 1	446.79	0.134	0.230	**0.447**
	pre 2	**171.85**	**0.321**	**0.469**	0.572
	pre 3	194.37	0.284	0.409	0.468
	pre 4	138.19	0.234	0.442	0.605
MovieAVG	base	> 1000	0.004	0.008	0.002
	pre 1	> 1000	0.007	0.019	0.002
	pre 2	> 1000	0.002	0.001	0.002
	pre 3	> 1000	0.003	0.001	0.002
	pre 4	> 1000	0.002	0.001	0.002
PureSVD	base	548.62	0.053	0.094	**0.631**
	pre 1	438.58	0.023	0.041	0.677
	pre 2	**138.56**	**0.418**	**0.567**	0.726
	pre 3	149.12	0.389	0.526	0.684
	pre 4	179.13	0.253	0.467	0.646
TopRated	base	701.59	0.007	0.009	**0.373**
	pre 1	786.79	0.006	0.009	0.407
	pre 2	212.39	0.215	0.367	0.397
	pre 3	212.39	**0.215**	**0.367**	0.397
	pre 4	**199.64**	0.147	0.303	**0.378**
LTR $\alpha = 0.9$	base	412.17	0.011	0.015	0.072

Tabella 7.4: risultati riassuntivi del precalcolo della URM su delicious

utenti dai gusti "difficili", approccio che si rivela vincente. L'algoritmo 3 invece di cambiare la scala dei rating, effettua la mediazione con la media dei rating dell'utente: quest'approccio si rivela essere il migliore per rendere la URM più densa, mantenendo inalterata la scala dei rating ed effettuando in ogni caso la mediazione per mezzo del fattore sociale α.

7.2.3 Epinions

Precalcolo della URM: variante 1 L'analisi delle recall ha mostrato che il preprocessing della matrice dei rating nella **variante 1** produce risultati peggiori rispetto alla versione originale. Anche le curve ROC non mostrano alcun miglioramento (per dettagli si vedano i grafici B.3a e B.3b in appendice).

Precalcolo della URM: variante 2 L'analisi della recall in figura 7.6 mostra come il preprocessing della matrice dei rating nella **variante 2** produce risultati leggermente migliori per gli algoritmi *COS*, *PureSVD* e *MovieAVG* (circa un incremento di 0.01 per ognuno *recall@10*), si veda la Tabella 7.5. L'analisi delle curve ROC in figura 7.7 non ha mostrato differenze sostanziali tra la versione sociale e la versione originale.

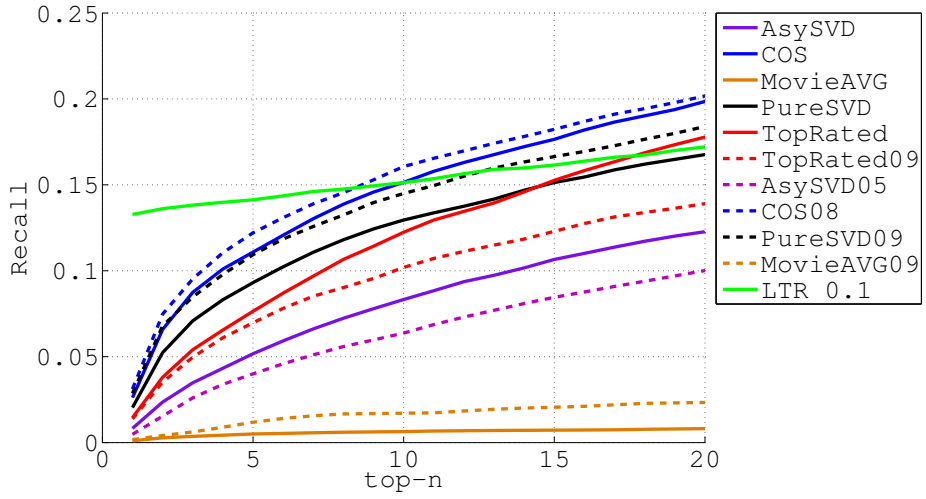


Figura 7.6: Epinions: Confronto Recall algoritmo 2

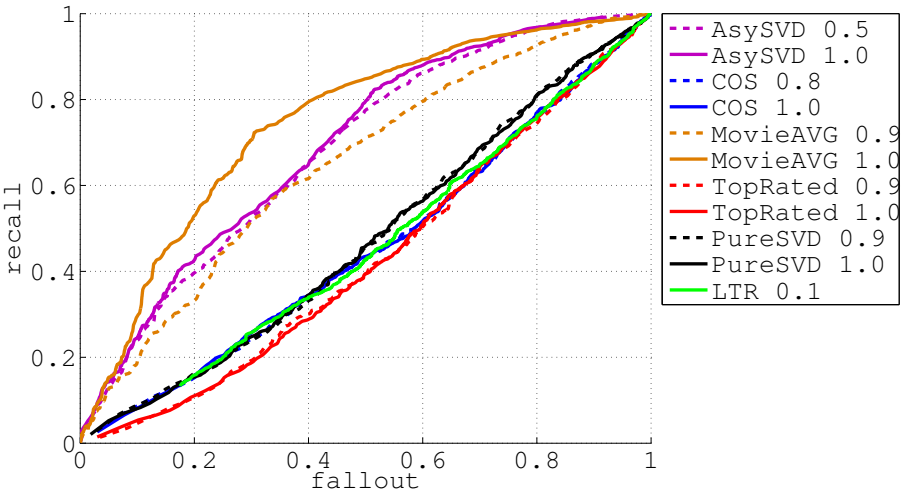


Figura 7.7: Epinions: Confronto curve ROC algoritmo 2

Precalcolo della URM: variante 3 L'analisi delle recall ha mostrato che il preprocessing della matrice dei rating nella **variante 3** produce risultati peggiori rispetto alla versione originale, tranne in *MovieAVG* in cui sebbene ci sia un miglioramento, si tratta di valori in assoluto bassi (Tabella 7.5). Le curve ROC mostrano un miglioramento di *AsySVD* e *MovieAVG* (per dettagli si vedano i grafici B.3e e B.3f in appendice).

Precalcolo della URM: variante 4 L'analisi della recall e delle curve di ROC mostra che l'impatto del preprocessing della matrice dei rating nella **variante 4** è pressocché inesistente (per dettagli si veda il grafico B.3h in appendice). Questo risultato è attribuibile al fatto che questa variante agisce solo nel caso in cui vi sia la presenza simultanea del rating dell'utente e degli amici, che nel dataset di *Epinions* si verifica nello **0.82%** dei casi (si veda la Tabella 7.2).

Discussione dei risultati Gli algoritmi di preprocessing non hanno avuto molta influenza in nessuno degli algoritmi di riferimento: ciò è dovuto al fatto che la media degli amici dell'utente è bassa (circa 20), e la sparsità della URM è molto alta, come descritto nella sezione 4.4). Durante la fase di preprocesing si è presentata molto frequentemente la situazione in cui l'utente possiede un rating su un item mentre nessuno dei suoi amici possiede rating sullo stesso item. In situazioni di grande sparsità dei dati, poi, gli algoritmi di preprocessing che rendono la URM più densa generano rating che non rispecchiano effettivamente le preferenze di un utente. Lo studio su questo dataset ha quindi dato prova che

Algoritmo	variante	ARP	MRR	Recall@10	Fallout@10
AsySVD	base	**483.80**	**0.041**	**0.086**	0.028
	pre 1	490.98	0.009	0.010	**0.002**
	pre 2	495.44	0.031	0.066	0.022
	pre 3	506.16	0.025	0.050	0.011
	pre 4	**483.70**	**0.041**	**0.086**	0.028
COS	base	**378.14**	0.089	0.155	0.196
	pre 1	456.72	0.070	0.112	**0.148**
	pre 2	439.27	**0.098**	**0.164**	0.213
	pre 3	454.90	0.076	0.123	0.188
	pre 4	**378.49**	0.089	0.155	0.200
MovieAVG	base	> 1000	0.004	0.004	0.001
	pre 1	> 1000	0.005	0.005	0.001
	pre 2	> 1000	0.006	0.010	0.003
	pre 3	> 1000	0.007	0.011	0.003
	pre 4	> 1000	0.004	0.004	0.001
PureSVD	base	**359.57**	0.074	0.132	0.165
	pre 1	389.97	0.075	0.118	**0.148**
	pre 2	361.99	**0.089**	**0.147**	0.185
	pre 3	389.22	0.078	0.121	0.177
	pre 4	**358.62**	0.074	0.132	0.166
TopRated	base	**475.81**	**0.062**	**0.127**	**0.226**
	pre 1	530.16	0.053	0.105	0.196
	pre 2	530.16	0.053	0.105	0.196
	pre 3	530.16	0.053	0.105	0.203
	pre 4	**475.81**	**0.062**	**0.127**	**0.226**
LTR $\alpha = 0.1$	base	454.35	0.143	0.151	0.193

Tabella 7.5: risultati riassuntivi del precalcolo della URM su epinions

gli algoritmi di preprocessing non sono efficaci in presenza di matrici di rating e di trust molto sparse.

7.3 Discussione

7.3.1 Considerazioni sui dataset

Sparsità dei dati L'informazione utilizzata per la raccomandazione si fonda principalmente sui dati estratti dai rating degli utenti, tuttavia la densità dei rating disponibili in sistemi commerciali è spesso inferiore all'1% [73]: questo tipo di problema viene spesso indicato come *sparsità dei dati* o *cold start problem*[74]. Il social preprocessing si è rivelato un buon approccio al problema della sparsità dei dati, utilizzare le relazioni di amicizia per aumentare la densità della rating matrix ha dato risultati particolarmente promettenti nel dataset di *Delicious*.

Lo studio condotto sui dati di *Epinions* ha mostrato che su matrici di rating e di trust particolarmente sparse l'impatto del preprocessing è pressocché trascurabile. In futuro potrebbe essere utile formalizzare questo limite in termini di densità delle matrici per stabilire a priori se sia sensato o meno applicare questa tecnica sui dati a disposizione.

L'esecuzione su un dataset più denso come *LastFM* ha rivelato che il metodo proposto ha un impatto considerevole sulla raccomandazione.

Rating function Come spiegato in precedenza, il peggioramento in termini di fallout dei risultati ottenuti su *LastFM* è attribuibile alla funzione utilizzata per convertire l'informazione implicita in rating; i risultati ottenuti con *Delicious* (in cui è stata utilizzata una funzione di rating più sensibile al valore del conteggio delle interazioni *user-item*) confermano in parte questa ipotesi.

7.3.2 Considerazioni sugli algoritmi

In generale, delle quattro varianti di social preprocessing proposte, solo due hanno dato risultati soddisfacenti: la *variante 2* e la *variante 3*. Il social preprocessing nella *variante 1* produce un numero di falsi negativi particolarmente alto, mentre la *variante 4* ha un'incidenza troppo bassa e risulta ininfluente per LastFM ed Epinions, producendo invece buoni risultati sul dataset di Delicious.

Le varianti 2 e 3 hanno migliorato gli algoritmi dello stato dell'arte *PureSVD*, *TopRated* e *COS* che già nella versione originale avevano dato risultati promettenti. Anche *AsySVD* ha beneficiato del social preprocessing, sebbene nella versione originale i risultati non sono stati particolarmente eclatanti.

Le esecuzioni di *Movie AVG* su tutti e tre i dataset hanno dato risultati di per se particolarmente scadenti (recall@50 inferiore a 0.1), per cui non è stato

possibile misurare l'impatto del social preprocessing su questo algoritmo; valori bassi di recall dipendono dal fatto che l'algoritmo non si presta ad essere valutato con una metrica di recall, piuttosto si tende ad utilizzare RMSE e MAE. In futuro si potrebbero effettuare simulazioni su altri dataset su cui sia sensata un'analisi in termini di RMSE, così da poter valutare l'impatto del social modeling su questo algoritmo.

7.3.3 Il ruolo del fattore sociale

L'analisi dei risultati del preprocessing ha mostrato che il fattore sociale α è di per sè irrilevante: nella maggior parte dei casi, infatti, come descritto nell'introduzione al capitolo, non viene utilizzato. Il peso maggiore è dato dal particolare algoritmo di preprocessing. In Figura 7.8 sono riportate le curve di recall degli algoritmi di social preprocessing che hanno prodotto risultati migliori (varianti 2 e 3). Si utilizza un colore diverso per ogni variante, e diversi tipi di tratto per evidenziare i diversi valori del parametro sociale. Si osservi come entrambe le varianti superino in termini di recall l'algoritmo *PureSVD* eseguito sulle matrici originali. Fissato un algoritmo di preprocessing, la variazione della componente sociale ha un impatto trascurabile, si noti come le curve di recall con $\alpha = 0.0$ e $\alpha = 0.9$ siano pressocché sovrapposte.

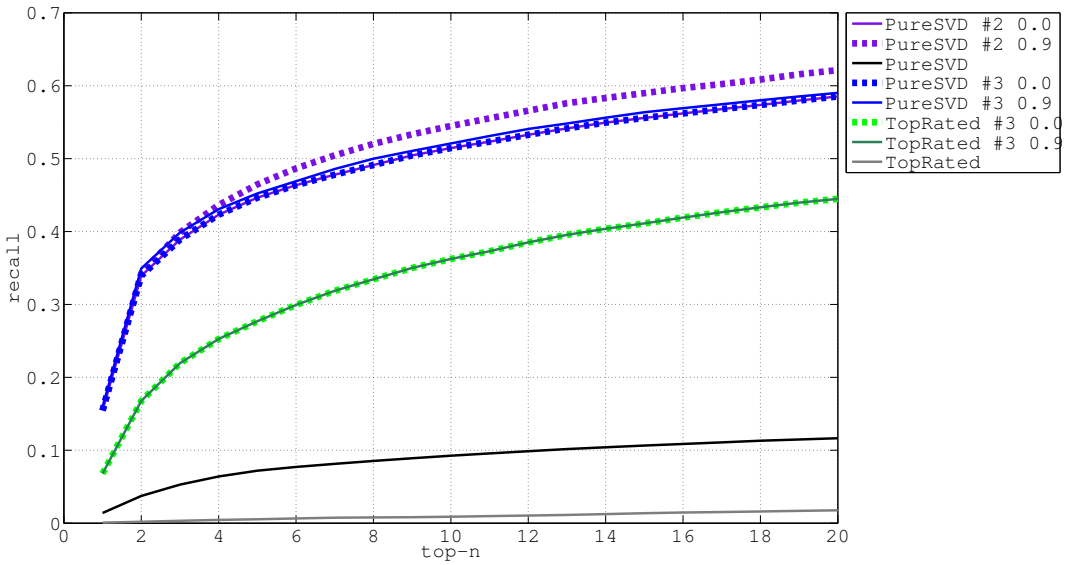


Figura 7.8: Delicious: variazione del fattore sociale

Capitolo 8

Conclusioni

Nel corso del nostro lavoro sono state analizzante diverse metodologie di integrazione delle informazioni sociali nell'ambito dei sistemi di raccomandazione. Sono stati identificati tre approcci di integrazione: *social preprocessing*, *social modeling* e *social postprocessing*. Mentre i primi due approcci sono stati approfonditi nella tesi, il *social postprocessing* non è stato trattato ed è stato lasciato a sviluppi futuri.

Un concetto fondamentale su cui verte la maggior parte del lavoro è il cosiddetto *fattore sociale*, cioè quel parametro che indica quanto in una decisione influisca la componente sociale rispetto al peso del profilo dell'utente.

In prima istanza si è cercato di migliorare l'algoritmo *Learning to recommend with social trust ensemble* [50] cercando di trovare una correlazione tra il parametro e il profilo sociale dell'utente. Si è modificato l'algoritmo originale al fine di calcolare un fattore sociale personalizzato per ogni utente durante la fase di training, quindi sono state impiegate tecniche di regressione lineare per poter fare inferenza sul parametro.

Questa strada si è rivelata poco promettente, infatti la variante proposta ha prodotto un miglioramento pressocché ininfluente; inoltre la regressione ha evidenziato la totale assenza di correlazione tra le variabili considerate (numero di amici e numero di rating) e il parametro sociale.

Riteniamo che comunque possa essere interessante scoprire quali siano le variabili che influenzano il fattore sociale ottimo per ogni utente. Una possibile strada potrebbe essere quella di contestualizzare la scelta dell'utente.

Quindi sono state esplorate strade alternative per integrare il fattore sociale nel modello (social modeling) modifcando algoritmi basati su metriche di ranking (CLiMF e ALSPR). La scelta di utilizzare algoritmi basati su informazioni im-

plicite è stata una conseguenza dell'analisi dei dataset pubblici a disposizione, dato che la maggior parte dell'informazione estraibile da un social network è di natura implicita (clickstream, pattern di navigazione, interazione con item). Le modifiche apportate hanno prodotto dei miglioramenti, seppur minimi, rispetto alle versioni originali degli algoritmi esaminati (circa +0.01 Recall@10).

L'esito del *social modeling* ha portato a pensare ad un differente approccio di inclusione delle informazioni sociali in un sistema di raccomandazione: il *social preprocessing*. Sono state sviluppate quattro diverse tecniche per preprocessare i dati in ingresso agli algoritmi collaborativi per includere le relazioni di amicizia. Tra queste, due si sono rivelate particolarmente efficaci, tanto da migliorare in termini di recall e ROC gli algoritmi collaborativi dello stato dell'arte. Il vero punto di forza del *social preprocessing* è quello di poter essere applicato ad un generico algoritmo collaborativo che riceve in ingresso una matrice dei rating, il che copre praticamente tutti gli algoritmi collaborativi.

Sviluppi futuri Alla luce degli esperimenti e dei risultati descritti, la ricerca condotta può essere estesa con la metodologia del *Social Postprocessing*, introdotta in Figura 1.1, ovvero quella di integrare, con una tecnica analoga al preprocessing, l'output di un generico algoritmo con l'informazione sociale derivata delle relazioni di amicizia.

Un altro possibile sviluppo consiste nell'applicare il *social preprocessing* a matrici di rating binari dedotte da informazioni implicite.

Sarebbe auspicabile effettuare altri test sui dataset di *LastFM* e *Delicious* utilizzando diverse *rating functions* per stabilire quanto influisca questo tipo di scelta.

Per dare maggiore significatività ai risultati ottenuti sarebbe inoltre opportuno utilizzare altre metodologie di test (possibilmente con un'effetiva interazione da parte dell'utente) e altre metriche di errore.

Appendice A

Notazione utilizzata

Nel mondo dei sistemi di raccomandazione possiamo modellare gli utenti come un insieme $U = \{u_1, \ldots, u_m\}$, mentre possiamo modellare l'insieme degli item, ovvero degli oggetti verso i quali l'utente esprime una preferenza come un insieme $I = \{i_1, \ldots, i_n\}$.

Le preferenze che un utente esprime verso un determinato item, che chiameremo *rating*, sarà denotato da $r_{u,i}$. Questo rating può essere qualsiasi numero reale ma, di solito, i rating sono numeri interi nell'intervallo [1,5]. In seguito ci occuperemo anche di rating impliciti: un *rating implicito* è un rating per il quale l'utente non ha espresso una vera e propria preferenza, ma la presenza del rating è stata derivata in modo implicito: $r_{u,i} > 0$ denota la presenza di un rating implicito. $r_{u,i} = 0$ denota, invece, l'assenza di rating, ovvero l'utente u non ha espresso nessuna preferenza, neanche implicita verso l'item i; questo significa che non si ha alcuna nozione sulla relazione tra l'utente u e l'item i.

In una *rete sociale* abbiamo anche le informazioni sugli amici che esprimono informazioni di fiducia, o *trust*, di un utente verso un altro utente, denotate da $t_{u,v}$ che possono essere binarie oppure variare con continuità nell'intervallo [0,1]; nei dataset presi in esame, comunque, le relazioni di amicizia sono binarie e simmetriche. Chiameremo con $RI_u = \{i_{u_1}, \ldots, i_{u_k}\}$ l'insieme degli item su cui l'utente u da espresso una preferenza; chiameremo invece l'insieme di utenti di cui u è amico con $TU_u = \{v \in U | t_{u,v} = 1\}$.

Il compito di un sistema di raccomandazione è il seguente: dato un utente u e un item i, predire il rating (sconosciuto) che l'utente darebbe a quell'item (questo è un problema di *predizione*).

Riducendo ulteriormente la complessità, si potrebbe pensare al problema della raccomandazione come: dato un utente si vuole generare una lista di item che potrebbero interessargli (questo è un problema di *ordinamento* o *ranking*). Ovviamente, predicendo il rating su tutti gli item, si potrebbero raccomandare gli

item più rilevanti all'utente, risolvendo quindi il problema dell'ordinamento: i due problemi, sebbene simili, sono differenti, ad esempio si può risolvere il problema del ranking senza risolvere quello della predizione.

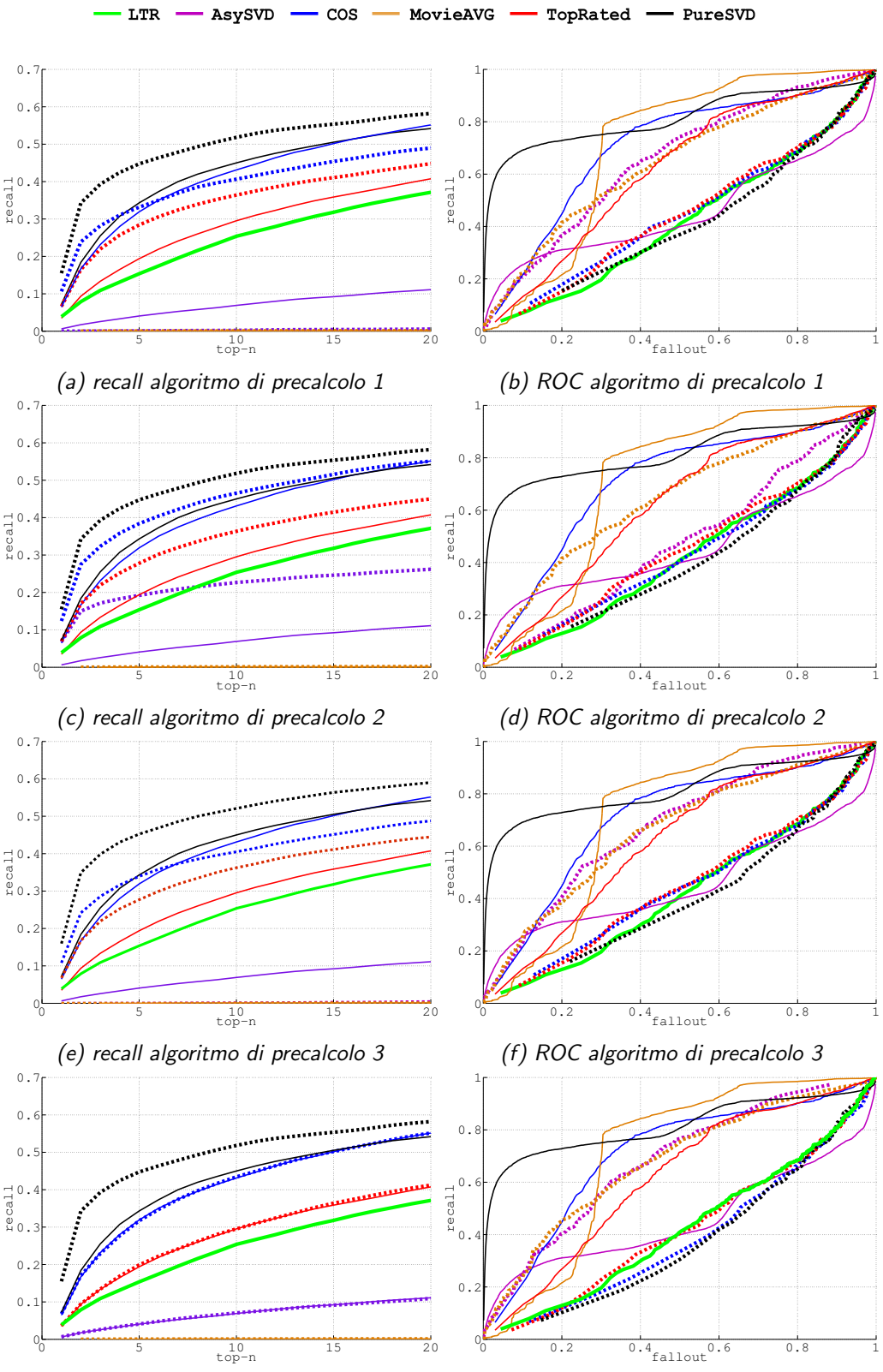
Appendice B

Grafici

In questo capitolo vengono riportati per completezza tutti i risultati delle simulazioni relative alla parte di *social preprocessing*, così da fornire tutte le informazioni necessarie per un'analisi esaustiva. Si mostrano per ognuno dei tre dataset i grafici di recall e le curve di ROC per ciascuna delle quattro varianti descritte nella sezione di *social preprocessing*. In ogni grafico è presente un confronto tra le versioni con e senza preprocessing dei seguenti algoritmi: AsySVD, COS, MovieAVG, TopRated e PureSVD; a questi è affiancato l'algoritmo sociale LTR per il confronto con lo stato dell'arte degli algoritmi sociali.

Si utilizza un colore diverso per ogni algoritmo, il tratto continuo si riferisce all'algoritmo nella versione non sociale, mentre il tratteggiato alla variante sociale. Per le varianti sociali è specificato il valore del parametro α usato per la simulazione. Ad esempio la didascalia *PureSVD 0.9* si riferisce all'algoritmo *PureSVD*, in versione sociale con preprocessing e con $\alpha = 0.9$, essendo sociale si utilizza un tratto discontinuo.

Nella prima colonna sono riportate le curve di recall. L'asse delle ascisse rappresenta la dimensione della lista di raccomandazione, l'asse delle ordinate il valore di recall della lista di raccomandazione di dimensione n. Nella seconda colonna si riportano le curve di ROC. Sull'asse delle ascisse è riportato il fallout, sull'asse delle ordinate la recall.



(a) recall algoritmo di precalcolo 1

(b) ROC algoritmo di precalcolo 1

(c) recall algoritmo di precalcolo 2

(d) ROC algoritmo di precalcolo 2

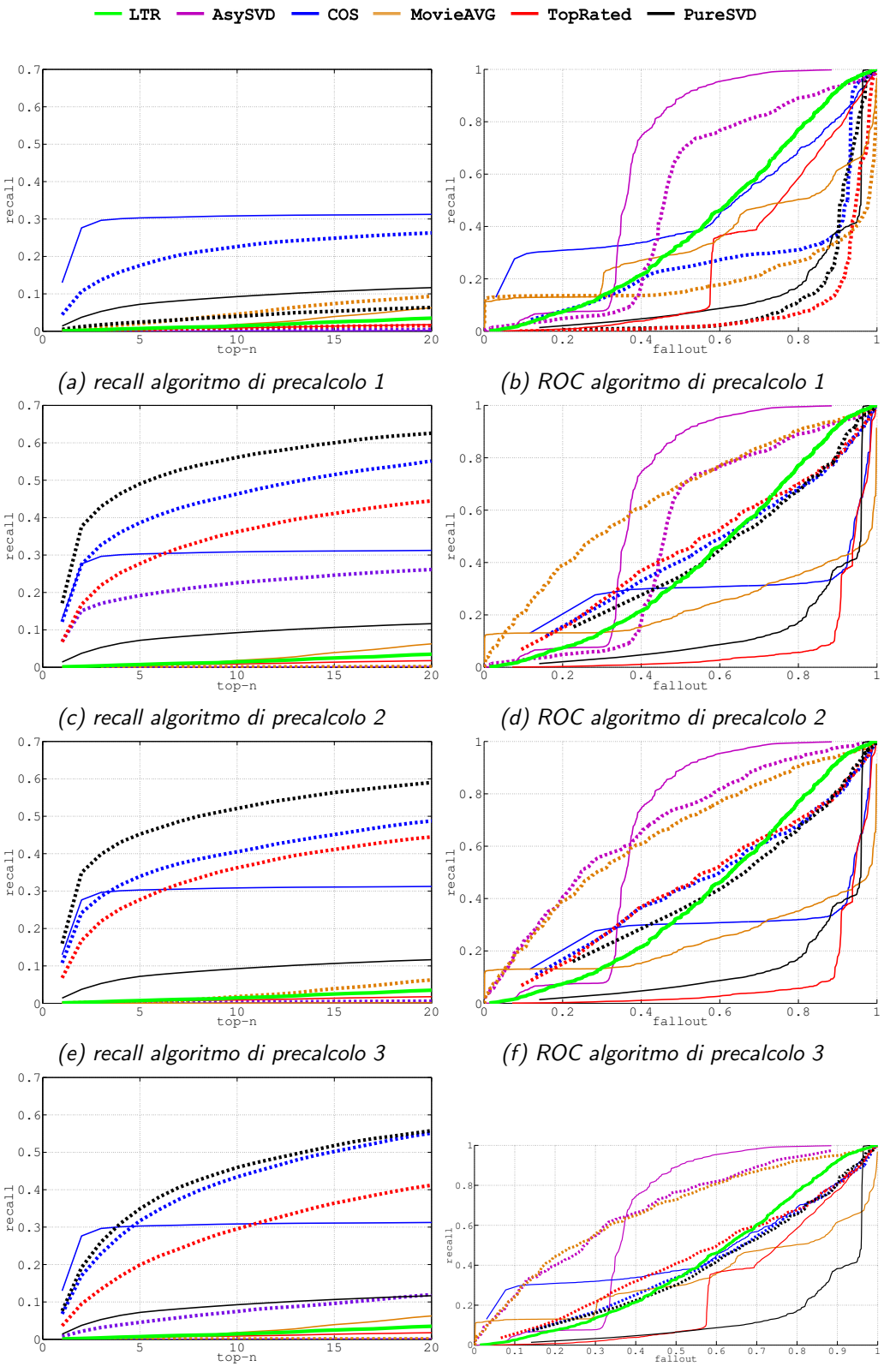
(e) recall algoritmo di precalcolo 3

(f) ROC algoritmo di precalcolo 3

(g) recall algoritmo di precalcolo 4

(h) ROC algoritmo di precalcolo 4

Figura B.1: Curve di ROC e recall per gli algoritmi di precalcolo sul dataset Last.FM



(a) recall algoritmo di precalcolo 1

(b) ROC algoritmo di precalcolo 1

(c) recall algoritmo di precalcolo 2

(d) ROC algoritmo di precalcolo 2

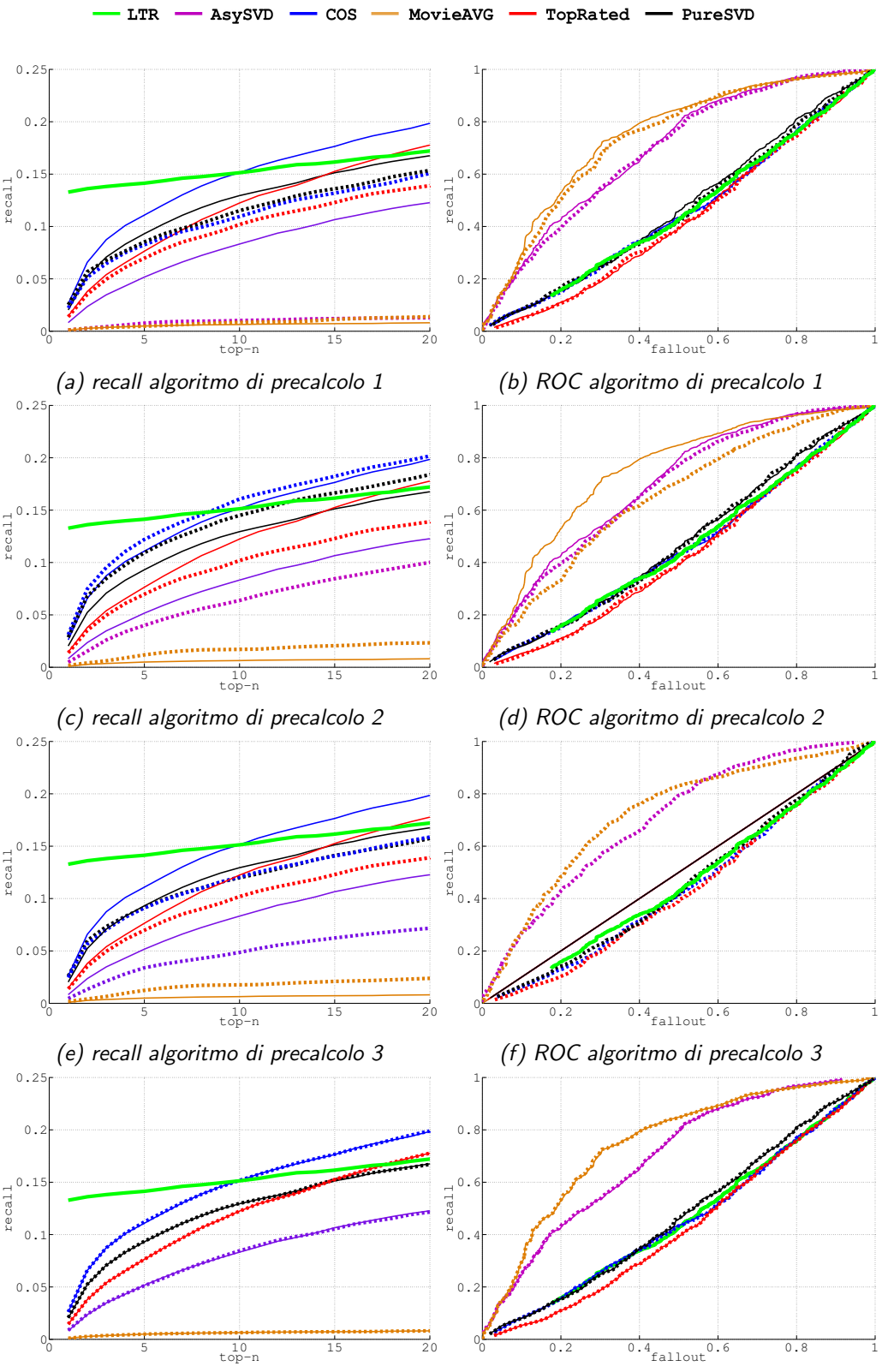
(e) recall algoritmo di precalcolo 3

(f) ROC algoritmo di precalcolo 3

(g) recall algoritmo di precalcolo 4

(h) ROC algoritmo di precalcolo 4

Figura B.2: Curve di ROC e recall per gli algoritmi di precalcolo sul dataset Delicious



(a) recall algoritmo di precalcolo 1

(b) ROC algoritmo di precalcolo 1

(c) recall algoritmo di precalcolo 2

(d) ROC algoritmo di precalcolo 2

(e) recall algoritmo di precalcolo 3

(f) ROC algoritmo di precalcolo 3

(g) recall algoritmo di precalcolo 4

(h) ROC algoritmo di precalcolo 4

Figura B.3: Curve di ROC e recall per gli algoritmi di precalcolo sul dataset Epinions

Bibliografia

[1] Gediminas Adomavicius e Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". In: *IEEE Trans. on Knowl. and Data Eng.* 17.6 (giu. 2005), pp. 734–749. ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.99. URL: http://dx.doi.org/10.1109/TKDE.2005.99.

[2] Asim Ansari, Skander Essegaier e Rajeev Kohli. "Internet Recommendation Systems". In: (2000), pp. 363–375.

[3] Lars Backstrom et al. "Group formation in large social networks: membership, growth, and evolution". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '06. Philadelphia, PA, USA: ACM, 2006, pp. 44–54. ISBN: 1-59593-339-5. DOI: 10.1145/1150402.1150412. URL: http://doi.acm.org/10.1145/1150402.1150412.

[4] Ricardo A. Baeza-Yates e Berthier Ribeiro-Neto. *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN: 020139829X.

[5] Marko Balabanović e Yoav Shoham. "Fab: content-based, collaborative recommendation". In: *Commun. ACM* 40.3 (mar. 1997), pp. 66–72. ISSN: 0001-0782. DOI: 10.1145/245108.245124. URL: http://doi.acm.org/10.1145/245108.245124.

[6] N K Bayma e A Ledbetterb. "Tunes that bind? predicting friendship strength in a music-based social network". In: *Information, Communication and Society* (2009), pp. 408–427.

[7] Nicholas J. Belkin e W. Bruce Croft. "Information filtering and information retrieval: two sides of the same coin?" In: *Commun. ACM* 35.12 (dic. 1992), pp. 29–38. ISSN: 0001-0782. DOI: 10.1145/138859.138861. URL: http://doi.acm.org/10.1145/138859.138861.

[8] James Bennett, Stan Lanning e Netflix Netflix. "The Netflix Prize". In: *In KDD Cup and Workshop in conjunction with KDD*. 2007.

[9] Daniel Billsus e Michael J. Pazzani. “Learning Collaborative Information Filters”. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 46–54. ISBN: 1-55860-556-8. URL: http://dl.acm.org/citation.cfm?id=645527.657311.

[10] Jeffrey D. Blume et al. *Estimation and Covariate Adjustment of ROC Curves and Underlying Test Score Distributions by*.

[11] T. Bogers e A. van den Bosch. “Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites”. In: (ott. 2009).

[12] John S. Breese, David Heckerman e Carl Kadie. “Empirical analysis of predictive algorithms for collaborative filtering”. In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. UAI’98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52. ISBN: 1-55860-555-X. URL: http://dl.acm.org/citation.cfm?id=2074094.2074100.

[13] Robin Burke. “Hybrid Recommender Systems: Survey and Experiments”. In: *User Modeling and User-Adapted Interaction* 12.4 (nov. 2002), pp. 331–370. ISSN: 0924-1868. DOI: 10.1023/A:1021240730564. URL: http://dx.doi.org/10.1023/A:1021240730564.

[14] Elica Campochiaro et al. “Do Metrics Make Recommender Algorithms?” In: *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops*. WAINA ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 648–653. ISBN: 978-0-7695-3639-2. DOI: 10.1109/WAINA.2009.127. URL: http://dx.doi.org/10.1109/WAINA.2009.127.

[15] Keunho Choi et al. “A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis.” In: *Electronic Commerce Research and Applications* 11.4 (2012), pp. 309–317. URL: http://dblp.uni-trier.de/db/journals/ecra/ecra11.html#ChoiYKS12.

[16] David Crandall et al. “Feedback effects between similarity and social influence in online communities”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’08. Las Vegas, Nevada, USA: ACM, 2008, pp. 160–168. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401914. URL: http://doi.acm.org/10.1145/1401890.1401914.

[17] Paolo Cremonesi, Yehuda Koren e Roberto Turrin. “Performance of recommender algorithms on top-n recommendation tasks”. In: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys ’10. Barcelona, Spain: ACM, 2010, pp. 39–46. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721. URL: http://doi.acm.org/10.1145/1864708.1864721.

[18] J. Delgado. *Memory-Based WeightedMajority Prediction for Recommender Systems*. 1999. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.7460.

[19] Joan DiMicco et al. “Motivations for social networking at work”. In: *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. CSCW ’08. San Diego, CA, USA: ACM, 2008, pp. 711–720. ISBN: 978-1-60558-007-4. DOI: 10.1145/1460563.1460674. URL: http://doi.acm.org/10.1145/1460563.1460674.

[20] Xiaowen Ding, Bing Liu e Philip S. Yu. “A holistic lexicon-based approach to opinion mining”. In: *Proceedings of the international conference on Web search and web data mining*. WSDM ’08. Palo Alto, California, USA: ACM, 2008, pp. 231–240. ISBN: 978-1-59593-927-2. DOI: 10.1145/1341531.1341561. URL: http://doi.acm.org/10.1145/1341531.1341561.

[21] Nicole B. Ellison. “Social Network Sites: Definition, History, and Scholarship”. In: *Journal of Computer-Mediated Communication* 13.1 (2007), pp. 210–230. ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.2007.00393.x. URL: http://dx.doi.org/10.1111/j.1083-6101.2007.00393.x.

[22] Jill Freyne et al. “Social networking feeds: recommending items of interest”. In: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys ’10. Barcelona, Spain: ACM, 2010, pp. 277–280. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864766. URL: http://doi.acm.org/10.1145/1864708.1864766.

[23] Susan Gauch et al. “The adaptive web”. In: a cura di Peter Brusilovsky, Alfred Kobsa e Wolfgang Nejdl. Berlin, Heidelberg: Springer-Verlag, 2007. Cap. User profiles for personalized information access, pp. 54–89. ISBN: 978-3-540-72078-2. URL: http://dl.acm.org/citation.cfm?id=1768197.1768200.

[24] L. Getoor e M. Sahami. “Using Probabilistic Relational Models for Collaborative Filtering”. In: *Working Notes of the KDD Workshop on Web Usage Analysis and User Profiling*. 1999.

[25] Ken Goldberg et al. “Eigentaste: A Constant Time Collaborative Filtering Algorithm”. In: *Inf. Retr.* 4.2 (lug. 2001), pp. 133–151. ISSN: 1386-4564. DOI: 10.1023/A:1011419012209. URL: http://dx.doi.org/10.1023/A:1011419012209.

[26] Georg Groh e Christian Ehmig. “Recommendations in taste related domains: collaborative filtering vs. social filtering”. In: *Proceedings of the 2007 international ACM conference on Supporting group work*. GROUP '07. Sanibel Island, Florida, USA: ACM, 2007, pp. 127–136. ISBN: 978-1-59593-845-9. DOI: 10.1145/1316624.1316643. URL: http://doi.acm.org/10.1145/1316624.1316643.

[27] Jianming He. “A social network-based recommender system”. AAI3437557. Tesi di dott. Los Angeles, CA, USA, 2010. ISBN: 978-1-124-37377-5.

[28] Jonathan L. Herlocker et al. “Evaluating collaborative filtering recommender systems”. In: *ACM Transactions on Information Systems* 22 (2004), pp. 5–53.

[29] Will Hill et al. “Recommending and evaluating choices in a virtual community of use”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 194–201. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223929. URL: http://dx.doi.org/10.1145/223904.223929.

[30] Thomas Hofmann. “Collaborative filtering via gaussian probabilistic latent semantic analysis”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. SIGIR '03. Toronto, Canada: ACM, 2003, pp. 259–266. ISBN: 1-58113-646-3. DOI: 10.1145/860435.860483. URL: http://doi.acm.org/10.1145/860435.860483.

[31] Thomas Hofmann. “Probabilistic latent semantic indexing”. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '99. Berkeley, California, United States: ACM, 1999, pp. 50–57. ISBN: 1-58113-096-1. DOI: 10.1145/312624.312649. URL: http://doi.acm.org/10.1145/312624.312649.

[32] Rong Hu e Pearl Pu. “Enhancing collaborative filtering systems with personality information”. In: *Proceedings of the fifth ACM conference on Recommender systems*. RecSys '11. Chicago, Illinois, USA: ACM, 2011, pp. 197–204. ISBN: 978-1-4503-0683-6. DOI: 10.1145/2043932.2043969. URL: http://doi.acm.org/10.1145/2043932.2043969.

[33] Yifan Hu, Yehuda Koren e Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets”. In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 263–272. ISBN: 978-0-7695-3502-9. DOI: 10.1109/ICDM.2008.22. URL: http://dx.doi.org/10.1109/ICDM.2008.22.

[34] M Jahrer e A Töscher. “Collaborative filtering ensemble for ranking”. In: *Proc. of KDD Cup Workshop at 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD ’11* (2011).

[35] Mohsen Jamali e Martin Ester. “A matrix factorization technique with trust propagation for recommendation in social networks”. In: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys ’10. Barcelona, Spain: ACM, 2010, pp. 135–142. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864736. URL: http://doi.acm.org/10.1145/1864708.1864736.

[36] Mohsen Jamali e Martin Ester. “TrustWalker: a random walk model for combining trust-based and item-based recommendation”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’09. Paris, France: ACM, 2009, pp. 397–406. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557067. URL: http://doi.acm.org/10.1145/1557019.1557067.

[37] Mohsen Jamali, Tianle Huang e Martin Ester. “A generalized stochastic block model for recommendation in social rating networks”. In: *Proceedings of the fifth ACM conference on Recommender systems*. RecSys ’11. Chicago, Illinois, USA: ACM, 2011, pp. 53–60. ISBN: 978-1-4503-0683-6. DOI: 10.1145/2043932.2043946. URL: http://doi.acm.org/10.1145/2043932.2043946.

[38] Robert Jäschke et al. “Tag recommendations in social bookmarking systems”. In: *AI Commun.* 21.4 (dic. 2008), pp. 231–247. ISSN: 0921-7126. URL: http://dl.acm.org/citation.cfm?id=1487691.1487696.

[39] Steve Jurvetson. “What exactly is Viral Marketing?” In: (). URL: http://currypuffandtea.files.wordpress.com/2008/03/viral-marketing.pdf.

[40] Diane Kelly e Jaime Teevan. “Implicit feedback for inferring user preference: a bibliography”. In: *SIGIR Forum* 37.2 (set. 2003), pp. 18–28. ISSN: 0163-5840. DOI: 10.1145/959258.959260. URL: http://doi.acm.org/10.1145/959258.959260.

[41] Heung-Nam Kim et al. “A semantic model for social recommender systems”. In: *Proceedings of the 23rd Canadian conference on Advances in Artificial Intelligence*. AI’10. Ottawa, Canada: Springer-Verlag, 2010, pp. 328–331. ISBN: 3-642-13058-5, 978-3-642-13058-8. DOI: 10.1007/978-3-642-13059-5_39. URL: http://dx.doi.org/10.1007/978-3-642-13059-5_39.

[42] Joseph A. Konstan et al. “GroupLens: applying collaborative filtering to Usenet news”. In: *Commun. ACM* 40.3 (mar. 1997), pp. 77–87. ISSN: 0001-0782. DOI: 10.1145/245108.245126. URL: http://doi.acm.org/10.1145/245108.245126.

[43] Yehuda Koren. “Factorization meets the neighborhood: a multifaceted collaborative filtering model”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’08. Las Vegas, Nevada, USA: ACM, 2008, pp. 426–434. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401944. URL: http://doi.acm.org/10.1145/1401890.1401944.

[44] Chuck Lam. “SNACK: incorporating social network information in automated collaborative filtering”. In: *Proceedings of the 5th ACM conference on Electronic commerce*. EC ’04. New York, NY, USA: ACM, 2004, pp. 254–255. ISBN: 1-58113-771-0. DOI: 10.1145/988772.988820. URL: http://doi.acm.org/10.1145/988772.988820.

[45] Ken Lang. “NewsWeeder: learning to filter netnews”. In: *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995, pp. 331–339. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.6286.

[46] Theodoros Lappas e Dimitrios Gunopulos. “Interactive recommendations in social endorsement networks”. In: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys ’10. Barcelona, Spain: ACM, 2010, pp. 127–134. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864735. URL: http://doi.acm.org/10.1145/1864708.1864735.

[47] Xin Li, Xin Su e Mengyue Wang. “Social network-based recommendation: a graph random walk kernel approach”. In: *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*. JCDL ’12. Washington, DC, USA: ACM, 2012, pp. 409–410. ISBN: 978-1-4503-1154-0. DOI: 10.1145/2232817.2232915. URL: http://doi.acm.org/10.1145/2232817.2232915.

[48] Nick Littlestone e Manfred K. Warmuth. “The weighted majority algorithm”. In: *Inf. Comput.* 108.2 (feb. 1994), pp. 212–261. ISSN: 0890-5401. DOI: 10.1006/inco.1994.1009. URL: http://dx.doi.org/10.1006/inco.1994.1009.

[49] Nathan N. Liu et al. “Wisdom of the better few: cold start recommendation via representative based rating elicitation”. In: *Proceedings of the fifth ACM conference on Recommender systems*. RecSys ’11. Chicago, Illinois, USA: ACM, 2011, pp. 37–44. ISBN: 978-1-4503-0683-6. DOI: 10.1145/2043932.2043943. URL: http://doi.acm.org/10.1145/2043932.2043943.

[50] Hao Ma, Irwin King e Michael R. Lyu. “Learning to recommend with social trust ensemble”. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’09. Boston, MA, USA: ACM, 2009, pp. 203–210. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1571978. URL: http://doi.acm.org/10.1145/1571941.1571978.

[51] Hao Ma et al. “SoRec: social recommendation using probabilistic matrix factorization”. In: *Proceedings of the 17th ACM conference on Information and knowledge management*. CIKM ’08. Napa Valley, California, USA: ACM, 2008, pp. 931–940. ISBN: 978-1-59593-991-3. DOI: 10.1145/1458082.1458205. URL: http://doi.acm.org/10.1145/1458082.1458205.

[52] Benjamin Marlin. “Modeling User Rating Profiles For Collaborative Filtering”. In: *Advances in Neural Information Processing Systems 16*. A cura di Sebastian Thrun, Lawrence Saul e Bernhard Schölkopf. Cambridge, MA: MIT Press, 2004.

[53] Paolo Massa e Paolo Avesani. “Trust-aware recommender systems”. In: *Proceedings of the 2007 ACM conference on Recommender systems*. RecSys ’07. Minneapolis, MN, USA: ACM, 2007, pp. 17–24. ISBN: 978-1-59593-730–8. DOI: 10.1145/1297231.1297235. URL: http://doi.acm.org/10.1145/1297231.1297235.

[54] Paolo Massa e Paolo Avesani. “Trust-aware recommender systems”. In: *Proceedings of the 2007 ACM conference on Recommender systems*. RecSys ’07. Minneapolis, MN, USA: ACM, 2007, pp. 17–24. ISBN: 978-1-59593-730–8. DOI: 10.1145/1297231.1297235. URL: http://doi.acm.org/10.1145/1297231.1297235.

[55] Nikolaos F. Matsatsinis e Andreas P. Samaras. “MCDA and preference disaggregation in group decision support systems”. In: *European Journal of Operational Research* 130.2 (2001), pp. 414–429. ISSN: 0377-2217. DOI:

10.1016/S0377-2217(00)00038-2. URL: http://www.sciencedirect.com/science/article/pii/S0377221700000382.

[56] Miller McPherson, Lynn Smith-Lovin e James M Cook. "Birds of a Feather: Homophily in Social Networks". In: *Annual Review of Sociology* 27.1 (2001), pp. 415–444. DOI: 10.1146/annurev.soc.27.1.415. eprint: http://arjournals.annualreviews.org/doi/pdf/10.1146/annurev.soc.27.1.415. URL: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.soc.27.1.415.

[57] Masahiro Morita e Yoichi Shinoda. "Information filtering based on user behavior analysis and best match text retrieval". In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '94. Dublin, Ireland: Springer-Verlag New York, Inc., 1994, pp. 272–281. ISBN: 0-387-19889-X. URL: http://dl.acm.org/citation.cfm?id=188490.188583.

[58] Reyn Nakamoto et al. "Tag-Based Contextual Collaborative Filtering". In: (2007). URL: http://www.iaeng.org/IJCS/issues_v34/issue_2/IJCS_34_2_08.pdf.

[59] Atsuyoshi Nakamura e Naoki Abe. "Collaborative Filtering Using Weighted Majority Prediction Algorithms". In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 395–403. ISBN: 1-55860-556-8. URL: http://dl.acm.org/citation.cfm?id=645527.657289.

[60] Tim O'Reilly. "What Is Web 2.0". In: *O'Reilly Network* (2005).

[61] Maciej Pacula. "A Matrix Factorization Algorithm for Music Recommendation using Implicit User Feedback". In: 2009.

[62] Dmitry Y. Pavlov e David M. Pennock. "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains". In: *In Proceedings of Neural Information Processing Systems*. MIT Press, 2002, pp. 1441–1448.

[63] Michael J. Pazzani. "A Framework for Collaborative, Content-Based and Demographic Filtering". In: *Artif. Intell. Rev.* 13.5-6 (dic. 1999), pp. 393–408. ISSN: 0269-2821. DOI: 10.1023/A:1006544522159. URL: http://dx.doi.org/10.1023/A:1006544522159.

[64] Michael Pazzani e Daniel Billsus. "Learning and Revising User Profiles: The Identification of Interesting Web Sites". In: *Mach. Learn.* 27.3 (giu. 1997), pp. 313–331. ISSN: 0885-6125. DOI: 10.1023/A:1007369909943. URL: http://dx.doi.org/10.1023/A:1007369909943.

[65] Al Mamunur Rashid et al. “Getting to know you: learning new user preferences in recommender systems”. In: *Proceedings of the 7th international conference on Intelligent user interfaces*. IUI ’02. San Francisco, California, USA: ACM, 2002, pp. 127–134. ISBN: 1-58113-459-2. DOI: 10.1145/502716.502737. URL: http://doi.acm.org/10.1145/502716.502737.

[66] Steffen Rendle e Lars Schmidt-Thieme. “Pairwise interaction tensor factorization for personalized tag recommendation”. In: *Proceedings of the third ACM international conference on Web search and data mining*. WSDM ’10. New York, New York, USA: ACM, 2010, pp. 81–90. ISBN: 978-1-60558-889-6. DOI: 10.1145/1718487.1718498. URL: http://doi.acm.org/10.1145/1718487.1718498.

[67] Paul Resnick et al. “GroupLens: an open architecture for collaborative filtering of netnews”. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. CSCW ’94. Chapel Hill, North Carolina, United States: ACM, 1994, pp. 175–186. ISBN: 0-89791-689-1. DOI: 10.1145/192844.192905. URL: http://doi.acm.org/10.1145/192844.192905.

[68] Francesco Ricci et al. *Recommender Systems Handbook*. 1st. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN: 0387858199, 9780387858197.

[69] Elaine Rich. “Readings in intelligent user interfaces”. In: a cura di Mark T. Maybury e Wolfgang Wahlster. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998. Cap. User modeling via stereotypes, pp. 329–342. ISBN: 1-55860-444-8. URL: http://dl.acm.org/citation.cfm?id=286013.286035.

[70] J. Rocchio. “Relevance Feedback in Information Retrieval”. In: *The SMART Retrieval System*. 1971, pp. 313–323. URL: http://scholar.google.com/scholar?hl=en%5C&%5C#38;lr=%5C&%5C#38;client=firefox-a%5C&%5C#38;q=relevance+feedback+in+information+retrieval%5C&%5C#38;btnG=Search.

[71] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0-201-12227-8.

[72] B. M. Sarwar et al. “Application of Dimensionality Reduction in Recommender Systems: A case study”. In: *WebKDD Workshop at the ACM SIGKKD*. 2000.

[73] Badrul Sarwar et al. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. WWW ’01. Hong Kong, Hong Kong: ACM, 2001, pp. 285–295.

ISBN: 1-58113-348-0. DOI: 10.1145/371920.372071. URL: http://doi.acm.org/10.1145/371920.372071.

[74] Andrew I. Schein et al. "Methods and metrics for cold-start recommendations". In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '02. Tampere, Finland: ACM, 2002, pp. 253–260. ISBN: 1-58113-561-0. DOI: 10.1145/564376.564421. URL: http://doi.acm.org/10.1145/564376.564421.

[75] Christian Schmitt, Dietmar Dengler e Mathias Bauer. "Multivariate preference models and decision making with the MAUT machine". In: *Proceedings of the 9th international conference on User modeling*. UM'03. Johnstown, PA, USA: Springer-Verlag, 2003, pp. 297–302. ISBN: 3-540-40381-7. URL: http://dl.acm.org/citation.cfm?id=1759957.1760006.

[76] Upendra Shardanand e Pattie Maes. "Social information filtering: algorithms for automating `word of mouth'". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223931. URL: http://dx.doi.org/10.1145/223904.223931.

[77] Sybil Shearin e Henry Lieberman. "Intelligent profiling by example". In: *Proceedings of the 6th international conference on Intelligent user interfaces*. IUI '01. Santa Fe, New Mexico, United States: ACM, 2001, pp. 145–151. ISBN: 1-58113-325-1. DOI: 10.1145/359784.360325. URL: http://doi.acm.org/10.1145/359784.360325.

[78] B. Sheth e P. Maes. "Evolving agents for personalized information filtering". In: mar. 1993, pp. 345–352. DOI: 10.1109/CAIA.1993.366590. URL: http://dx.doi.org/10.1109/CAIA.1993.366590.

[79] Yue Shi et al. "CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering". In: *Proceedings of the sixth ACM conference on Recommender systems*. RecSys '12. Dublin, Ireland: ACM, 2012, pp. 139–146. ISBN: 978-1-4503-1270-7. DOI: 10.1145/2365952.2365981. URL: http://doi.acm.org/10.1145/2365952.2365981.

[80] Rashmi R. Sinha e Kirsten Swearingen. "Comparing Recommendations Made by Online Systems and Friends". In: *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*. 2001.

[81] Gábor Takács e Domonkos Tikk. “Alternating least squares for personalized ranking”. In: *Proceedings of the sixth ACM conference on Recommender systems*. RecSys ’12. Dublin, Ireland: ACM, 2012, pp. 83–90. ISBN: 978-1-4503-1270-7. DOI: 10.1145/2365952.2365972. URL: http://doi.acm.org/10.1145/2365952.2365972.

[82] Junichi Tatemura. “Virtual reviewers for collaborative exploration of movie reviews”. In: *Proceedings of the 5th international conference on Intelligent user interfaces*. IUI ’00. New Orleans, Louisiana, United States: ACM, 2000, pp. 272–275. ISBN: 1-58113-134-8. DOI: 10.1145/325737.325870. URL: http://doi.acm.org/10.1145/325737.325870.

[83] Loren Terveen et al. “PHOAKS: a system for sharing recommendations”. In: *Commun. ACM* 40.3 (mar. 1997), pp. 59–62. ISSN: 0001-0782. DOI: 10.1145/245108.245122. URL: http://doi.acm.org/10.1145/245108.245122.

[84] Karen H. L. Tso-Sutter, Leandro Balby Marinho e Lars Schmidt-Thieme. “Tag-aware recommender systems by fusion of collaborative filtering algorithms”. In: *Proceedings of the 2008 ACM symposium on Applied computing*. SAC ’08. Fortaleza, Ceara, Brazil: ACM, 2008, pp. 1995–1999. ISBN: 978-1-59593-753-7. DOI: 10.1145/1363686.1364171. URL: http://doi.acm.org/10.1145/1363686.1364171.

[85] L. Ungar e D. Foster. “Clustering Methods For Collaborative Filtering”. In: *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California, 1998. URL: citeseer.ist.psu.edu/ungar98clustering.html.

[86] Vishvas Vasuki et al. “Affiliation recommendation using auxiliary networks”. In: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys ’10. Barcelona, Spain: ACM, 2010, pp. 103–110. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864731. URL: http://doi.acm.org/10.1145/1864708.1864731.

[87] Frank Edward Walter, Stefano Battiston e Frank Schweitzer. “A model of a trust-based recommendation system on a social network”. In: *Autonomous Agents and Multi-Agent Systems* 16.1 (feb. 2008), pp. 57–74. ISSN: 1387-2532. DOI: 10.1007/s10458-007-9021-x. URL: http://dx.doi.org/10.1007/s10458-007-9021-x.

[88] Robert Wetzker, Winfried Umbrath e Alan Said. “A hybrid approach to item recommendation in folksonomies”. In: *Proceedings of the WSDM ’09 Workshop on Exploiting Semantic Annotations in Information Retrieval*. ESAIR ’09. Barcelona, Spain: ACM, 2009, pp. 25–29. ISBN: 978-1-60558-

430-0. DOI: 10.1145/1506250.1506255. URL: http://doi.acm.org/10.1145/1506250.1506255.

[89] Christopher C. Yang, Hsinchun Chen e Kay Hong. "Visualization of large category map for internet browsing". In: *Decis. Support Syst.* 35.1 (apr. 2003), pp. 89–102. ISSN: 0167-9236. DOI: 10.1016/S0167-9236(02)00101-X. URL: http://dx.doi.org/10.1016/S0167-9236(02)00101-X.

[90] Quan Yuan, Li Chen e Shiwan Zhao. "Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation". In: *Proceedings of the fifth ACM conference on Recommender systems*. RecSys '11. Chicago, Illinois, USA: ACM, 2011, pp. 245–252. ISBN: 978-1-4503-0683-6. DOI: 10.1145/2043932.2043975. URL: http://doi.acm.org/10.1145/2043932.2043975.

[91] Q. Yuan et al. "Augmenting Collaborative Recommender by Fusing Explicit Social Relationships". In: *ACM RecSys'09 Workshop on Recommender Systems & the Social Web*. New York, ott. 2009.

[92] Yi Zhang, Jamie Callan e Thomas Minka. "Novelty and redundancy detection in adaptive filtering". In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '02. Tampere, Finland: ACM, 2002, pp. 81–88. ISBN: 1-58113-561-0. DOI: 10.1145/564376.564393. URL: http://doi.acm.org/10.1145/564376.564393.

[93] Elena Zheleva, Hossam Sharara e Lise Getoor. "Co-evolution of social and affiliation networks". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. Paris, France: ACM, 2009, pp. 1007–1016. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557128. URL: http://doi.acm.org/10.1145/1557019.1557128.

[94] Xujuan Zhou et al. "The state-of-the-art in personalized recommender systems for social networking". In: *Artif. Intell. Rev.* 37.2 (feb. 2012), pp. 119–132. ISSN: 0269-2821. DOI: 10.1007/s10462-011-9222-1. URL: http://dx.doi.org/10.1007/s10462-011-9222-1.

[95] Li Zhuang, Feng Jing e Xiao-Yan Zhu. "Movie review mining and summarization". In: *Proceedings of the 15th ACM international conference on Information and knowledge management*. CIKM '06. Arlington, Virginia, USA: ACM, 2006, pp. 43–50. ISBN: 1-59593-433-2. DOI: 10.1145/1183614.1183625. URL: http://doi.acm.org/10.1145/1183614.1183625.

[96] M Zuckerberg et al. "Dynamically providing a news feed about a user of a social network". In: (2010).