UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea magistrale in
Informatica

IL TITOLO DELLA TESI

Relatore: Relatore 1
Correlatore: Correlatore 1

Tesi di Laurea di:
Lorenzo D'Alessandro
Matr. Nr. 939416

ANNO ACCADEMICO 2020-2021

Dedica

Ringraziamenti

Questa sezione, facoltativa, contiene i ringraziamenti.

Indice

Capitolo 1

Introduzione

Introduzione...

1.1 I contenuti

Spiegazione problema...

1.2 Organizzazione della tesi

Organizzazione tesi...

Capitolo 2

Stato dell'arte

I recommender system sono algoritmi mirati a suggerire oggetti rilevanti agli utenti. La definizione di oggetto è generica e include ad esempio film da guardare, libri da leggere, prodotti da comprare o qualsiasi altra cosa a seconda del contesto in cui sono implementati. Quando gli utenti interagiscono con il sistema generano dei feedback. Questi feedback possono essere di due tipi: espliciti o impliciti. I feedback espliciti sono valori numerici che un utente assegna ad un prodotto, i feedback impliciti riflettono indirettamente le opinioni di un utente osservando la cronologia degli acquisti, i link clickati, gli elementi visualizzati, etc. Basandosi sui feedback passati, i RS imparano un modello per prevedere quanto un utente può essere interessato a nuovi oggetti. Questi oggetti sono poi ordinati in base alla pertinenza prevista per l'utente. In ultimo, gli oggetti con il rank più alto vengono suggeriti all'utente. La relazione tra utenti e oggetti è rappresentata con una matrice R_M in cui sono memorizzati i rating passati degli utenti. La *ratings matrix* è definita come:

$$R_M : U \times I \to R$$

dove $U = \{u_1, \ldots, u_m\}$ rappresenta l'insieme degli utenti, $I = \{i_1, \ldots, i_n\}$ rappresenta l'insieme degli utenti, e $R = \{r_1, \ldots, r_k\}$ rappresenta l'insieme dei possibili ratings che un utente ha espresso riguardo a degli oggetti. Un valore mancante nella ratings matrix può avere due significati: l'utente non vuole esprimere un'opinione su un oggetto specifico, oppure l'utente non conoscendo ancora l'oggetto non può averlo valutato [1].

I recommender system si dividono principalmente in 3 categorie:

- collaborative filtering
- content-based
- hybrid

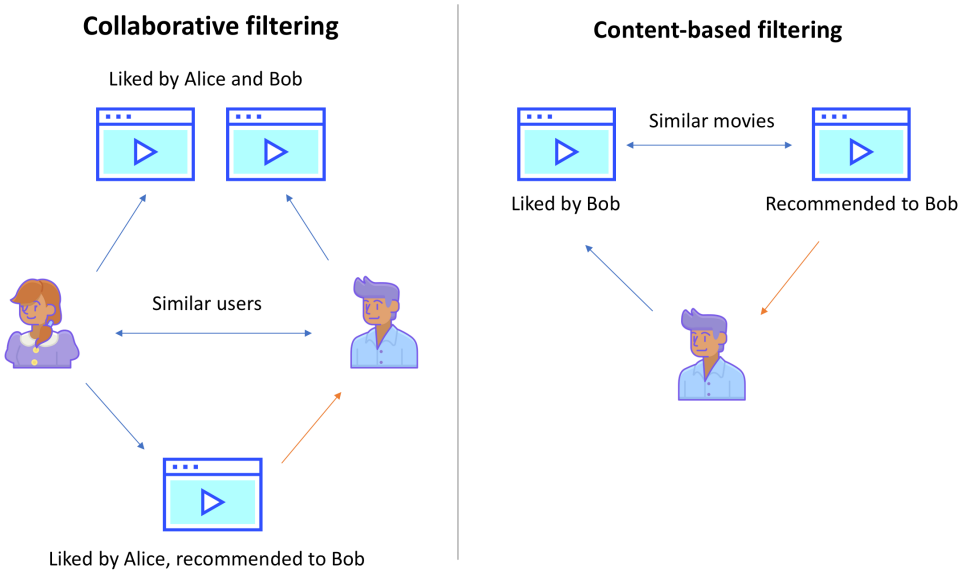

Figura 1: Collaborative filtering vs Content-based

In figura Figura 1 è schematizzato il modo diverso in cui operano i metodi collaborative filtering rispetto a quelli content-based.

2.1 Collaborative filtering recommender system

Collaborative filtering (CF) è considerata la tecnica di raccomandazione più popolare e ampiamente utilizzata nei RS. Il presupposto alla base di CF è che le persone con preferenze simili valuteranno gli stessi oggetti con ratings simili. CF quindi sfrutta le informazioni sul comportamento passato o le opinioni di una comunità di utenti esistente per prevedere quali elementi potranno piacere o saranno interessanti per l'utente corrente del sistema [2]. Gli approcci CF puri non sfruttano né richiedono alcuna conoscenza degli oggetti stessi ma solo dei feedback degli utenti.

La classe di algoritmi più famosa è quella di Matrix Factorization, ma recentemente sono stati sviluppati diversi approcci basati sul deep learning [3].

2.1.1 Vantaggi e svantaggi

Vantaggi:

- *Nessuna conoscenza del dominio necessaria:* il modello può funzionare in domini in cui non c'è molto contenuto associato agli oggetti e in cui il contenuto è difficile da analizzare per un sistema informatico (come opinioni e ideali) [4].

- *Serendipity:* il modello ha la capacità di fornire consigli fortuiti, il che significa che può consigliare elementi pertinenti per l'utente anche senza che il contenuto si trovi nel profilo dell'utente permettendo all'utente di scoprire nuovi interessi [4] [5].

Svantaggi:

- *Problema del cold-start:* Si riferisce alla situazione in cui il sistema di raccomandazione non ha abbastanza informazioni su un utente od un oggetto per poter fare previsioni rilevanti. Un nuovo oggetto inserito nel RS di solito non ha voti, ed è quindi improbabile che venga raccomandato. Un oggetto non consigliato passa inosservato a gran parte della community. Il problema è presente anche per i nuovi utenti: gli utenti che hanno espresso nessuna o poche valutazioni non ricevono raccomandazioni affidabili [6].

- *Problema di data sparsity:* questo è il problema che si verifica a causa della mancanza di informazioni sufficienti, cioè quando solo pochi rispetto al numero totale di oggetti disponibili in un database sono valutati dagli utenti. Ciò porta ad una rating matrix sparsa, e raccomandazioni poco efficaci [4].

- *Scalabilità:* Questo è un altro problema associato agli algoritmi di raccomandazione perchè il tempo di computazione cresce linearmente con il numero di utenti e oggetti. Un sistema di raccomandazione efficiente con un dataset limitato potrebbe non esserlo quando il volume è incrementato [4].

2.1.2 Matrix factorization

Gli algoritmi basati su matrix factorization(MF) caratterizzano sia utenti che oggetti mediante dei vettori di fattori estratti dai pattern sui ratings. Una corrispondenza alta tra i fattori di un utente e un oggetto porta ad una raccomandazione. Questi metodi sono diventati popolari negli ultimi anni perchè combinano scalabilità e accuratezza.

Più formalmente, i modelli basati su matrix factorization mappano utenti e oggetti in uno spazio di fattori latenti di dimensionalità f, tale che le interazioni tra utenti e oggetti sono modellate come prodotti in quello spazio. Di conseguenza, ogni oggetto i è associato con un vettore $q_i \in \mathbb{R}^f$, e ogni utente u con un vettore $p_u \in \mathbb{R}^f$. Per un dato oggetto i, gli elementi di q_i indicano la misura in cui l'oggetto

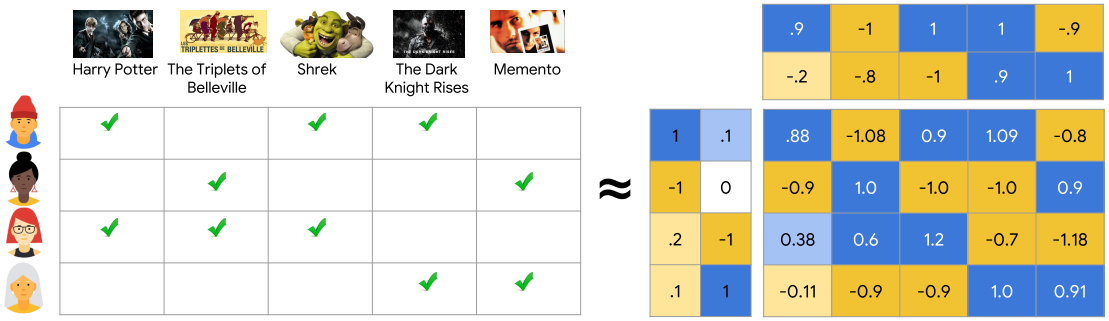

Figura 2: Approssimazione matrice dei ratings con matrix factorization

possiede quei fattori, positivi o negativi. Per un dato utente u, gli elementi di p_u indicato l'entità dell'interesse che l'utente ha per gli oggetti che hanno un valore alto sui fattori corrispondenti, positivi o negativi. Il prodotto scalare $q_i^T p_u$ indica l'interesse dell'utente u per le caratteristiche dell'oggetto i [7]. Quindi il rating r_{ui} può essere approssimato come

$$r_{ui} = q_i^T p_u \tag{1}$$

Il problema principale è calcolare il mapping di ogni oggetto e utente su dei vettori $q_i, p_u \in \mathbb{R}^f$. Una volta che il recommender system ha completato il mapping, può facilmente stimare il rating che un utente darà a qualsiasi oggetto utilizzando l'equazione 1.

In Figura 3 è mostrato come la matrice dei ratings $A \in \mathbb{R}^{m \times n}$, con m numero di utenti e n numero di oggetti, viene decomposta in due matrici di dimensionalità minore:

- Una matrice di embedding per gli utenti $P \in \mathbb{R}^{m \times f}$, in cui la riga i è l'embedding dell'utente i
- Una matrice di embedding per gli oggetti $Q \in \mathbb{R}^{n \times f}$, in cui la colonna j è l'embedding dell'utente j

Gli embedding di P e Q sono imparati in modo tale che il prodotto PQ^T sia una buona approssimazione della matrice dei ratings A

2.2 Content-based recommender system

Nei recommender system content-based (CB), gli attributi descrittivi degli oggetti sono usati per produrre raccomandazioni. Il termine "content" indica queste

descrizioni. Nei metodi content-based, i ratings degli utenti sono combinati con le informazioni disponibili per gli oggetti, per poi essere usati come training data per creare un modello di classificazione o regressione specifico per l'utente. Questo modello specifico dell'utente viene utilizzato per prevedere se alla persona corrispondente piacerà un articolo per il quale la sua valutazione è ancora sconosciuta [8].

Mentre nei CF la similarità tra due oggetti (o due utenti) è calcolato come la correlazione o la similarità tra i ratings forniti dagli altri utenti, i recommender system content-based sono progettati per consigliare oggetti simili a quelli che l'utente ha preferito in passato. Non considerando gli altri utenti, la lista di raccomandazioni può essere generata anche se c'è un solo utente.

2.2.1 Vantaggi e svantaggi

Vantaggi:

- *Consigliare nuovi oggetti:* Questi modelli hanno la capacità di consigliare nuovi oggetti anche se non ci sono valutazioni fornite dagli utenti a differenza dei modelli collaborative filtering [4].
- ???

Svantaggi:

- *Features degli oggetti:* la precisione del modello dipende dall'insieme delle features che descrivono gli oggetti. Identificare le features più rilevanti non è semplice e dipende molto dall'applicazione specifica [1].
- *Content overspecialization:* Dato che i metodi CB si affidano solo alle caratteristiche degli oggetti già valutati dall'utente corrente, egli riceverà solo raccomandazioni simili ad altri oggetti già definiti nel suo profilo [4].
- *Dimensione training set:* In modo da consentire a un RS content-based di imparare le preferenze di un utente, egli dovrà valutare un numero sufficiente di oggetti. In caso contrario, il RS non avrà sufficienti informazioni per imparare un modello accurato e fallirà nel raccomandare oggetti a utenti con pochi o nessun ratings.

2.3 Hybrid recommender system

I modelli ibridi combinano tipi diversi di sistemi di raccomandazione per formare dei modelli in grado di superare le debolezze dei modelli singoli. In [8] sono descritti tre modi per creare recommender system ibridi:

1. *Ensemble design:* Con questo metodo i risultati degli algoritmi base sono combinati in un output singolo più robusto. Il principio fondamentale è molto simile ai metodi di ensemble usati in molte applicazioni di data mining come clustering, classificazione e analisi degli outlier. Gli ensemble design possono essere formalizzati nel modo seguente. Sia R^k una matrice $m \times n$ contenente le predizioni di m utenti per n oggetti dell'algoritmo k-esimo, con $k \in \{1, \ldots, q\}$. Pertanto, un totale di q algoritmi diversi sono usati per ottenere queste predizioni. L'elemento (u, j)-esimo di R^k contiene il rating predetto per l'utente u sull'oggetto j dall'algoritmo k-esimo. Gli elementi della matrice originale R sono replicati in ogni R^k, e solo gli elementi non presenti in R variano nei differenti R^k a causa dei diversi risultati degli algoritmi. Il risultato finale è ottenuto combinando le predizioni $R^1, \ldots, R^q$ in un singolo output. La combinazione può essere fatta in vari modi, ad esempio calcolando la media pesata delle varie predizioni. Le caratteristiche comuni di questi algoritmi sono usare sistemi di raccomandazione già esistenti e produrre uno score/ranking unico.

2. *Monolithic design:* In questo caso, viene creato un algoritmo di raccomandazione integrato utilizzando vari tipi di dati. A volte non esiste una chiara distinzione tra le varie parti (es. content-based e collaborative filtering) dell'algoritmo. In altri casi, potrebbe essere necessario modificare gli algoritmi algoritmi di raccomandazione esistenti per essere usati all'interno dell'approccio generale, anche quando c'è una chiara distinzione tra gli algoritmi utilizzati. Pertanto, questo approccio tende a integrare più strettamente le varie fonti di dati e non è possibile visualizzare facilmente i singoli componenti come black-boxes separate.

3. *Mixed systems:* Come per gli ensembles, questi sistemi usano diversi algoritmi di raccomandazione come black-boxes, ma gli oggetti raccomandati dai vari sistemi sono presentati insieme senza essere combinati.

2.4 Context-aware recommender system

I recommender system context-aware (CARS) producono le loro raccomandazioni utilizzando informazioni addizionali che definiscono la situazione di un utente. Queste informazioni addizionali sono chiamate *contesto*.
Alcuni esempi di contesto sono:

1. *Data e ora:* Dalle informazioni di data e ora è possibile estrarre diverse features contestuali come il momento della giornata, il giorno della settimana, week-end, vacanze, stagioni ed altro ancora. Una raccomandazione

potrebbe essere rilevante la mattina ma non il pomeriggio, e viceversa. Le raccomandazioni sui vestiti invernali o estivi possono essere molto diverse.

2. *Posizione:* Con la crescente popolarità del GPS disponibile ormai su qualunque telefono, le raccomandazioni sensibili alla posizione dell'utente hanno guadagnato importanza. Per esempio, un viaggiatore potrebbe desiderare raccomandazioni su ristoranti vicini alla propria posizione. Questo può essere fatto aggiungendo la posizione come contesto nel recommender system.

3. *Informazioni sociali:* Il contesto sociale è spesso importante per un sistema di raccomandazione. Le informazioni su amici, tag e cerchie sociali di un utente possono avere un impatto sul processo di raccomandazione. Per esempio una persona potrebbe scegliere di guardare un film diverso a seconda che lo guardi con i suoi genitori o con i suoi amici.

Il contesto può essere ottenuto in vari modi [9] che includono:

1. *Esplicitamente* ponendo domande dirette alle persone rilevanti o richiedendo le informazioni con altri mezzi. Per esempio, un sito web potrebbe ottenere informazioni contestuali chiedendo agli utenti di compilare un form.

2. *Implicitamente* dai dati o dall'ambiente, come il cambio di posizione rilevato da una compagnia di telefonia mobile. In questo caso non è necessario fare nulla in termini di interazione con l'utente o altre fonti di informazioni contestuali perché l'informazione contestuale è acceduta direttamente e i dati sono estratti da essa.

3. *Per inferenza* usando metodi statistici o di data mining.

2.4.1 Approccio multidimensionale

Il problema tradizionale di raccomandazione può essere visto come apprendere una funzione che associa le coppie utente-oggetto ai ratings. La funzione corrispondete f_R è definita come:

$$f_R : U \times I \rightarrow rating$$

Quindi la rating function mappa da uno spazio bidimensionale di utenti e oggetti ai ratings. I CARS generalizzano questo approccio utilizzando un approccio multidimensionale in cui la ratings function può essere vista come un mapping da una matrice n-dimensionale all'insieme dei ratings [1].

$$g_R : D_1 \times D_2 \cdots \times D_n \rightarrow rating$$

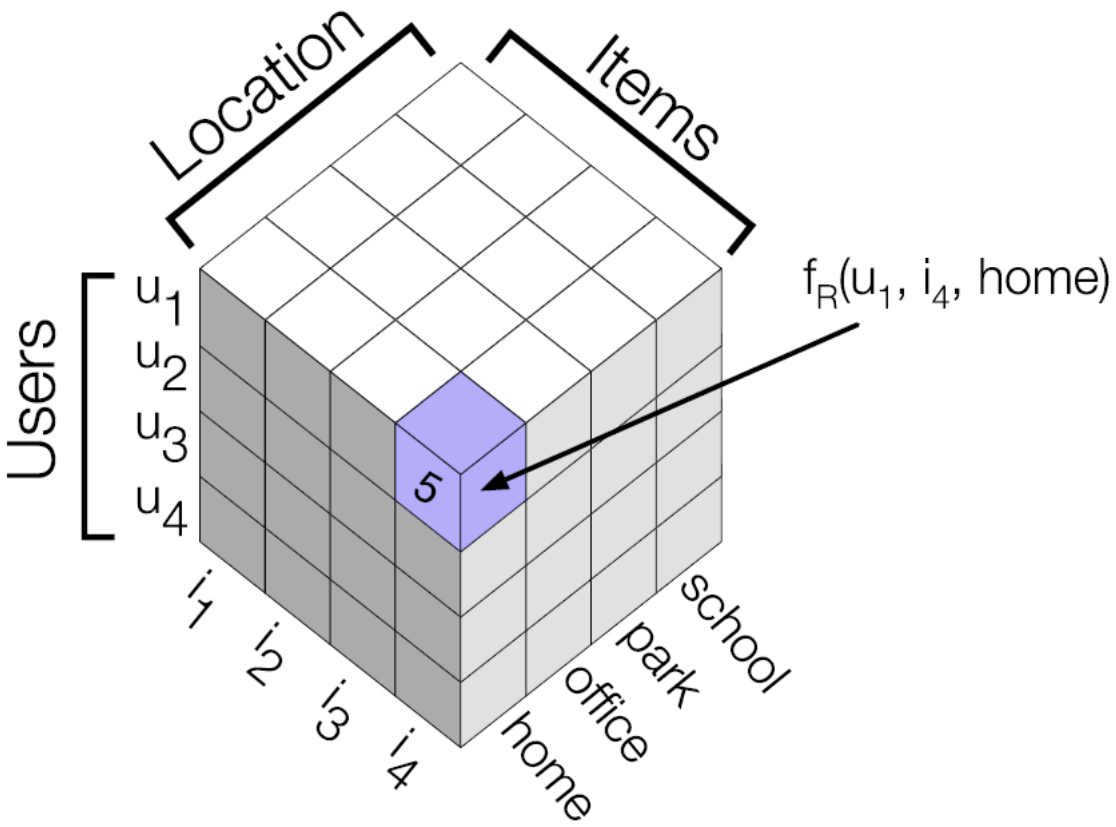

Figura 3: Esempio di un cubo multidimensionale per $User \times Item \times Location$

In questo caso, il risultato è un cubo n-dimensionale al posto che una matrice bi-dimensionale. Le diverse dimensioni sono denotate come $D_1 \dots D_n$. Due di queste dimensioni saranno sempre utenti e oggetti, le altre D_i dimensioni corrispondono alle features del contesto [8]. Il contesto può essere applicato nelle varie fasi del processo di raccomandazione. Si possono identificare tre categorie principali:

1. *Contextual pre-filtering:* In questo paradigma, le informazioni riguardo il contesto attuale sono utilizzate per selezionare o costruire l'insieme di dati rilevanti (la matrice dei ratings). Poi i ratings mancanti dai dati selezionati possono essere predetti utilizzando qualsiasi sistema di raccomandazione 2D tradizionale.

2. *Contextual post-filtering:* In questo paradigma, le informazioni contestuali sono inizialmente ignorate, e i ratings sono predetti utilizzando qualsiasi sistema di raccomandazione 2D traidizionale. Poi, l'insieme di raccomandazioni viene regolato utilizzando le informazioni di contesto.

3.

Contextual post-filtering (or contextualization of recommendation output). In this recommendation paradigm (presented in Figure 7.4b), contextual information is initially ignored, and the ratings are predicted using any traditional 2D 7 Context-Aware Recommender Systems 233 recommender system on the entire data. Then, the resulting set of recommendations is adjusted (contextualized) for each user using the contextual information

Capitolo 3

Classificatore

Capitolo 4

Datasets

Capitolo 5

Risultati

Capitolo 6

Conclusioni

6.1 Conclusioni

Conclusioni...

6.2 Sviluppi futuri

Sviluppi futuri...

Bibliografia

[1] Mattia G. Campana and Franca Delmastro. Recommender systems for online and mobile social networks: A survey. *Online Social Networks and Media*, 3-4:75–97, 2017.

[2] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, USA, 1st edition, 2010.

[3] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system. *ACM Computing Surveys*, 52(1):1–38, Feb 2019.

[4] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[5] Google Developers. Collaborative filtering advantages and disadvantages. `https://developers.google.com/machine-learning/recommendation/collaborative/summary`, 2020.

[6] Jesùs Bobadilla, Fernando Ortega, Antonio Hernando, and Jesùs Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[8] Charu C. Aggarwal. *Recommender Systems - The Textbook*. Springer, 2016.

[9] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.