Knowledge Management and Protection Systems (KMaPS)

Corso di Laurea in Ingegneria

Part 7a: Recommender Systems

Daniele Cenni, daniele.cenni@unifi.it

Seminario per il costo KMAPS del Prof. Paolo Nesi

DISIT Lab http://www.disit.dinfo.unifi.it/

Department of Information Engineering
Distributed Systems and Internet Technology Lab
Via S. Marta 3 - 50139 Firenze, Italy

http://www.disit.org

Recommender System (RS)

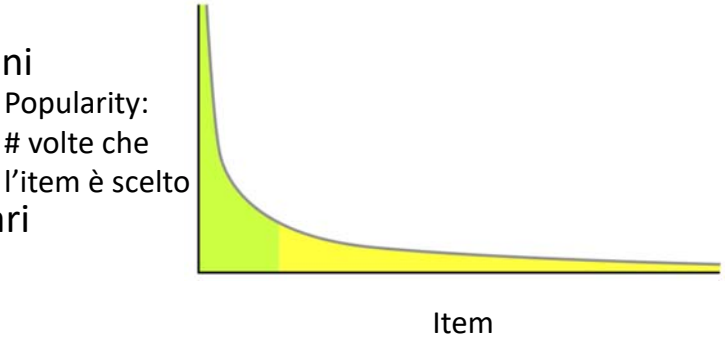
- È uno strumento software che fornisce suggerimenti ad un utente
  - I suggerimenti possono essere oggetti da acquistare, musica da ascoltare, notizie da leggere ecc.
  - In generale si parla di *item*
- È un sistema che cerca di predire il giudizio che un utente darebbe a un item o ad una persona che non ha ancora avuto modo di valutare direttamente
  - Si utilizza un modello, costruito sulla base delle caratteristiche dell'item da valutare (*content-based approach*) oppure sulla base del contesto sociale (*collaborative filtering approach*)
- Le tecniche utilizzate fanno ricorso a data mining, information retrieval, statistica, machine learning

Applicazioni

- Prodotti
  - Amazon, i suggerimenti tengono conto delle scelte fatte da utenti simili a quello considerato
- Notizie
  - Suggeriscono notizie all'utente che sono considerate interessanti da utenti simili
  - La similarità è basata sull'analisi del testo, o dal fatto che le notizie sono state lette da utenti simili
- Film/Musica
  - Netflix, le raccomandazioni sono basate sui giudizi espressi dagli utenti
  - Spotify
    - Since you follow ---, we recommend --- for your collection
    - You listened to ---. Here's an album you might like
- Articoli scientifici
  - ScienceDirect, article suggestion
    - *These articles have key terms similar to those in the article you downloaded*

Long Tail Phenomenon

- Chris Anderson (2004): https://www.wired.com/2004/10/tail
- I negozi fisici, al contrario di quelli online, possono offrire soltanto i prodotti più richiesti
  - Lo spazio a disposizione negli scaffali è limitato
  - Una libreria di città può offrire poche migliaia di libri, Amazon ne offre milioni
  - Non possono fornire raccomandazioni
  - Offrono soltanto gli item nella parte verde della curva
  - Cut-off: area verde = area gialla, si genera profitto con gli item meno popolari
- I servizi online traggono vantaggio dal fatto di avere un'offerta completa
  - Dispongono di grandi quantità di dati
  - Fanno business su entrambe le aree della curva
  - Sono costretti a creare sistemi di raccomandazione
    - Non possono presentare all'utente tutta la lista dei possibili item perché sono troppi



Netflix Prize

- 2006, Netflix lancia un contest per cercare di migliorare il proprio sistema di raccomandazioni
  - http://www.netflixprize.com
- 1 M$ a chi fosse stato capace di migliorare del 10% il proprio algoritmo (CineMatch)
  - 17,000 film
  - 500,000 utenti
  - 100 milioni di valutazioni
- RMSE (Root Mean Square Error), metrica prescelta per valutare l'efficacia dell'algoritmo
  - CineMatch: RMSE = 0.9525
  - Obbiettivo: RMSE <= 0.8572
- 2009, BellKor's Pragmatic Chaos vince il contest proponendo un algoritmo che migliora RMSE del 10.06% (0.8567)
- 2012, Netflix nonostante tutto non lo utilizza
  - "we evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment"
  - http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

Recommender System (RS)

- Deve fornire buone raccomandazioni e mostrare informazioni sintetiche ed esaustive ad esse relative
- Le raccomandazioni devono essere
  - Facili da leggere (si deve capire a colpo d'occhio di cosa si tratta)
  - Facili da valutare ("*non mi piace*", "*l'ho già comprato*")
    - Il sistema di rating deve essere facile da comprendere (stelline)
- Può essere
  - *Generico*, tutti gli utenti ricevono le stesse raccomandazioni
  - *Demografico*, tutti gli utenti nella stessa categoria ricevono le stesse raccomandazioni
  - *Contestuale*, le raccomandazioni dipendono soltanto dall'attività corrente
  - *Persistente*, le raccomandazioni dipendono da interessi consolidati

Strumenti

- Machine Learning, Statistica
- Modelli Off-line (tengono conto delle caratteristiche consolidate e globali del sistema)
- Modelli On-line (tengono conto delle componenti dinamiche)
- Click-through rate (CTR): numero di utenti che cliccano/numero di utenti totali
- Costruzione del profilo utente (e.g. Natural Language Processing per comprendere i contenuti)

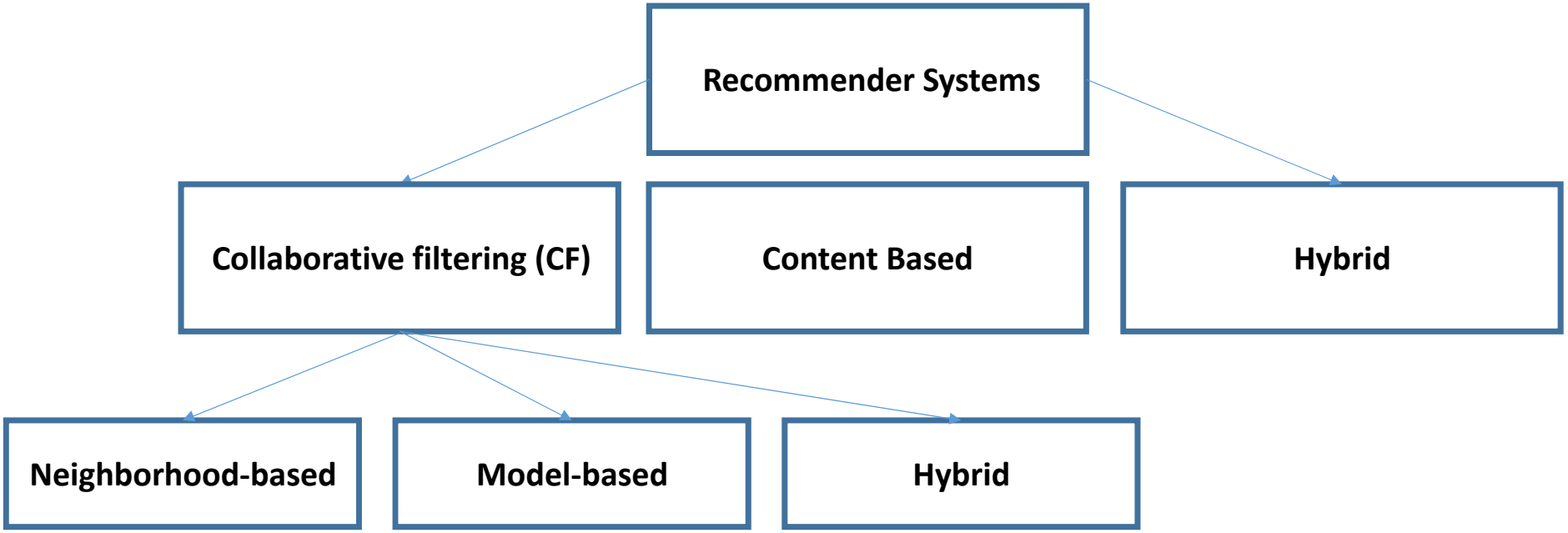
Workflow

- Generalmente ci si basa sulla conoscenza pregressa di utenti e contenuti simili
- L'utente *i* visita l'item *j* e fornisce in modo diretto (rating) o indiretto (click/no-click) una valutazione y_{ij}
- RS fornisce raccomandazioni (predice il rating di item non osservati) sulla base delle caratteristiche x_i dell'utente *i* e di y_{ij}
- x_i, dati demografici, browse history, search history
  - Il profile dell'utente

Classificazione RS

- ***Content-based***, analizzano le caratteristiche degli item già valutati positivamente dagli utenti e ne suggeriscono di simili
- ***Collaborative filtering (CF)***, forniscono raccomandazioni sulla base di misure di similarità fra gli utenti o gli item
  - Raccomandano ad un utente gli item che sono preferiti dagli utenti simili
  - Cold start user
- ***Ibridi***, combinano le due tecniche precedenti

Classificazione RS



Utility Matrix

- RS è definito come una funzione (utility function)
  - C: Customers, I: Items, R: Ratings

$$u: C \times I \rightarrow R$$

	I_1	I_2	…	…	I_n
U_1	4	?	1		
U_2		2	3	4	
…					
U_n		3		6	1

- Gli spazi vuoti corrispondono a item non valutati
- Che giudizio darà U_1 di I_2?

Utility Matrix

- Generalmente molti elementi della matrice sono vuoti
  - Si parla di matrici *sparse*
  - Gli utenti forniscono valutazioni soltanto per un numero limitato di item
- L'obbiettivo di un RS è di predire i valori degli spazi vuoti della matrice
  - Non è necessario predirli tutti
    - Non è interessante raccomandare item con valutazioni mediocri
- RS raccomanda un numero esiguo di item (quelli con rating più elevato)

Content-based

- Analizzano le caratteristiche degli item in modo da raccomandare item che sono simili ad item che l'utente ha già valutato positivamente, o che sono compatibili con il suo profilo
- Esempi
  - Un utente Netflix che ha visto molti film con lo/a stesso/a attore/attrice, regista, genere (e.g., western, commedia, comico)
    - Il sistema cercherà di suggerire film dello stesso tipo
  - Siti di notizie o blog
    - Il sistema cercherà di suggerire notizie o blog con contenuti simili
    - La similarità è valutata sulla base dell'analisi del testo

Item Profile

- Nei sistemi content-based si costruisce un profilo, cioè una collezione di record che rappresentano caratteristiche peculiari di quell'item (*feature*)
- Esempio: un film
  - Gli attori, le attrici
  - Il regista
  - L'anno di produzione
  - Il genere
  - L'autore della colonna sonora
  - …
- Ma ci sono altri tipi di item per i quali le feature non sono così evidenti
  - Come distinguere il topic di un articolo o di un blog o più in generale di una web page?
  - Si eliminano le stop words (articoli, pronomi, congiunzioni ecc.) dal documento e poi si calcola tf-idf
  - Si considerano come feature le *n* parole con il più alto tf-idf, che sono quelle rappresentative del documento (n può essere fisso, una percentuale del totale delle parole), o quelle sopra una certa soglia

Item Profile

- Un *item profile* è un insieme di caratteristiche (*feature*) che descrivono un item
  - Contestuali, e.g., attori/attrici, regista, genere, titolo
  - Testuali, insieme di parole rilevanti
    - Si usa tf-idf (**t**erm **f**requency times **i**nverse **d**ocument **f**requency)
- tf-idf
  - È una metrica che descrive l'importanza di una parola in un documento
  - Il profilo di un documento è costituito dall'insieme delle parole con la più alta tf-idf

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

  - $\text{tf}_{t,d}$, numero di volte che il termine t compare nel document d
  - idf_t, $log \frac{N}{df_t}$ dove df_t è il numero di documenti che contengono il termine t ed N è il numero totale di documenti

Item features da tag

- E per le immagini?
- Sono descritte da pixel, non hanno un contenuto semantico
- Possiamo chiedere all'utente di mettere un *tag* (etichetta) all'immagine, e questo vale per ogni tipo di contenuto
- Uno dei primi esempi di tagging di pagine web è il sito del.icio.us, poi comprato da Yahoo!
  - L'obbiettivo era di costruire un sistema di ricerca, l'utente scrive una query e il sistema restituisce come risultati le pagine che erano state etichettate con le parole della query
  - Ovviamente i tag possono essere usati per fornire raccomandazioni: se un utente richiede spesso pagine con certi tag, allora si possono suggerire pagine con gli stessi tag
  - Problema: questo sistema funziona solo se l'utente ha fornito dei tag, e se i tag corretti sono tanti in rapporto a quelli errati

Collaborative Filtering

- L'utente riceve raccomandazioni sulla base delle valutazioni passate di tutti gli utenti
- User-based collaborative filtering
  - Dato un utente U, trovare l'insieme degli utenti D le cui valutazioni sono simili a quelle dell'utente U
  - Utilizzare i rating forniti dagli utenti D per fornire raccomandazioni all'utente U
- Item-based collaborative filtering
  - Costruire una matrice di item in modo da determinare le relazioni fra coppie di item
  - Utilizzare la matrice per predire le preferenze dell'utente

Sistemi ibridi

- Usano sistemi di raccomandazione con tecniche diverse e ne combinano i risultati
- Integrano un sistema di tipo collaborative filtering con le caratterische proprie di uno content-based
  - Tengono conto dell'item profile (utile nel caso di nuovi item)
  - Tengono conto del profilo demografico dell'utente (utile nel caso di nuovi utenti)

Rating

- È un valore numerico o non numerico assegnato da un utente a un item/utente
- È il valore che viene scritto nella utility matrix
- Il popolamento della matrice ha a che fare con la costruzione del profilo utente, che può essere implicito o esplicito
  - ***Implicito***, si spia il comportamento dell'utente che interagisce con il sistema. Se l'utente guarda un film, legge un articolo, compra un oggetto, possiamo dedurre che ha espresso una preferenza: la matrice conterrà un 1 in corrispondenza di quell'item, altrimenti 0
  - ***Esplicito***, ci si basa sulle valutazioni fornite direttamente dall'utente. È l'approccio utilizzato da Netflix
    - Può essere numeric o semantico (tag, label)
  - ***Ibrido***, combina i due sistemi precedenti
- È spesso arrotondato (3, 4, 5 diventano 1), normalizzato (si sottrae la media), nel caso di calcolo di misure di similarità

Similarità: Jaccard

- Jaccard Index (Jaccard Similarity Coefficient), misura la similarità di due insiemi

$$j(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Distanza di Jaccard, misura la dissimilarità (è il complemento della precedente)

$$J_\delta(A, B) = 1 - j(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

  - Tiene conto solo del numero di item per i quali è stata fornita una valutazione
  - È appropriata quando la utility matrix consiste solo di 1 e di spazi vuoti

Similarità: Jaccard (esempio)

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4			5	1	
U_2	5	5	4			
U_3				2	4	5
U_4		4	1	3		

similarità

$$j(U_1, U_2) = \frac{|U_1 \cap U_2|}{|U_1 \cup U_2|} = \frac{1}{5} = 0.2$$

distanza

$$J_\delta(U_1, U_2) = \frac{|U_1 \cup U_2| - |U_1 \cap U_2|}{|U_1 \cup U_2|} = \frac{5-1}{5} = \frac{4}{5} = 0.8$$

$$j(U_1, U_3) = \frac{|U_1 \cap U_3|}{|U_1 \cup U_3|} = \frac{2}{4} = 0.5$$

$$J_\delta(U_1, U_3) = \frac{|U_1 \cup U_3| - |U_1 \cap U_3|}{|U_1 \cup U_3|} = \frac{4-2}{4} = \frac{2}{4} = 0.5$$

- U_1 e U_2 sono molto distanti (0.8)
- U_1 sembra più vicino a U_3 che a U_2
- Però U_1 e U_3 sono in disaccordo su I_4 e I_5, mentre U_1 e U_2 sostanzialmente concordano su I_1
- Jaccard non è una buona misura di similarità per valori diversi da 1 della matrice

Similarità: cosine distance

- Gli item sono rappresentati come vettori nello spazio degli utenti
- La similarità è rappresentata dal coseno dell'angolo fra i due vettori
- Dati due vettori di attributi A e B, è definita come

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2}\sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

- Funziona anche per valori diversi da 1 della utility matrix
- Solitamente si normalizzano i valori della matrice sottraendo la media (per riga, colonna o entrambe), prima di calcolarla

Cosine distance (esempio)

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4			5	1	
U_2	5	5	4			
U_3				2	4	5
U_4		4	1	3		

$$\cos(\theta_{U_1,U_2}) = \frac{U_1 \cdot U_2}{\|U_1\|\|U_2\|} = \frac{4 \cdot 5}{\sqrt{4^2+5^2+1^2}\sqrt{5^2+5^2+4^2}} = 0.380$$

$$\cos(\theta_{U_1,U_3}) = \frac{U_1 \cdot U_3}{\|U_1\|\|U_3\|} = \frac{5 \cdot 2 + 4 \cdot 1}{\sqrt{4^2+5^2+1^2}\sqrt{2^2+4^2+5^2}} = 0.322$$

- Gli spazi vuoti valgono 0
- Un coseno (positivo) più elevato implica un angolo più piccolo, e quindi una minore distanza
- U_1 è più vicino a U_2 che a U_3

Rounding (esempio)

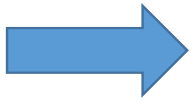
	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4			5	1	
U_2	5	5	4			
U_3				2	4	5
U_4		4	1	3		

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	1			1		
U_2	1	1	1			
U_3					1	1
U_4		1		1		

- Consideriamo i rating pari a 3, 4, 5 come se fossero tutti 1, e quelli inferiori a 3 come non presenti
- $J_\delta(U_1, U_2) = {}^3/_4$
- $J_\delta(U_1, U_3) = 1$
- U_1 è più vicino a U_2 che a U_3

Normalization (esempio)

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4			5	1	
U_2	5	5	4			
U_3				2	4	5
U_4		4	1	3		



	I_1	I_2	I_3	I_4	I_5	I_6
U_1	2/3			5/3	-7/3	
U_2	1/3	1/3	-2/3			
U_3				-5/3	1/3	4/3
U_4		4/3	-5/3	1/3		

- Si sottrae a ogni rating dell'utente la media delle sue valutazioni
- $\cos(\theta_{U_1,U_2}) = \frac{U_1 \cdot U_2}{\|U_1\|\|U_2\|} = \frac{(2/3)\cdot(1/3)}{\sqrt{(2/3)^2+(5/3)^2+(-7/3)^2}\sqrt{(1/3)^2+(1/3)^2+(-2/3)^2}} = 0.092$
- $\cos(\theta_{U_1,U_3}) = \frac{U_1 \cdot U_3}{\|U_1\|\|U_3\|} = \frac{(5/3)\cdot(-5/3)+(-7/3)\cdot(1/3)}{\sqrt{(2/3)^2+(5/3)^2+(-7/3)^2}\sqrt{(-5/3)^2+(1/3)^2+(4/3)^2}} = -0.559$
- U_1 e U_3 sono più lontani di U_1 e U_2 e nessuna coppia è molto vicina
- Risultato ragionevole, visto che U_1 e U_3 sono in disaccordo su I_4 e I_5, mentre U_1 e U_2 concordano sostanzialmente su I_1

Similarità: Pearson correlation

- Coefficiente di correlazione di Pearson

$$P_{a,u} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r_a})(r_{u,i} - \bar{r_u})}{\sqrt{\sum_{i \in I}((r_{a,i} - \bar{r_a})^2 \sum_{i \in I}(\sum_{i \in I}((r_{a,i} - \bar{r_a})^2)^2}}$$

dove

- I è l'insieme degli item valutati da entrambi gli utenti
- $r_{u,i}$ è la valutazione data dall'utente *u* sull'item *I*
- $\bar{r_a}, \bar{r_u}$ sono le valutazioni medie date dagli utenti *a* ed *u*

Utility matrix: calcolare i valori

- Usando la user similarity: si trovano gli n utenti più simili a U e si fa una media dei loro rating per l'item I (ovviamente considerando soltanto quelli che hanno espresso un rating)
  - Per calcolare il valore per l'utente U e l'item I si normalizza la matrice
    - Per ognuno degli n utenti si sottrae il loro rating medio dal loro rating per l'item i-esimo
    - Si fa la media della differenza fra questi utenti che hanno espresso un rating
    - Si aggiunge la media al rating medio che U dà a tutti gli item
- Usando la item similarity: si trovano gli m item più simili a I e si fa una media dei loro rating dati da U
- Non è sufficiente trovare un solo valore della matrice, per poter fare raccomandazioni è necessario stimare tutti gli elementi della riga relativa all'utente U, o almeno stimare la maggior parte degli elementi vuoti

Metriche di valutazione

- ***True positive rate*** *TP/(TP+FN)*, la frazione di casi positivi correttamente classificati come appartenenti alla classe positiva
- ***False negative rate*** *FN/(TP+FN)*, la frazione di casi positivi erroneamente classificati come appartenenti alla classe negativa
- ***False positive rate*** *FP/(FP+TN)*, la frazione di casi negativi erroneamente classificati come appartenenti alla classe positiva
- ***True negative rate*** *TN/(FP+TN)*, la frazione di casi negativi correttamente classificati come appartenenti alla classe negativa

TP = true positive (hit), TN = true negative (correct rejection), FP = false positive (false alarm, type I error), FN = false negative (miss, type II error)

Metriche di valutazione

- **Predictive accuracy**

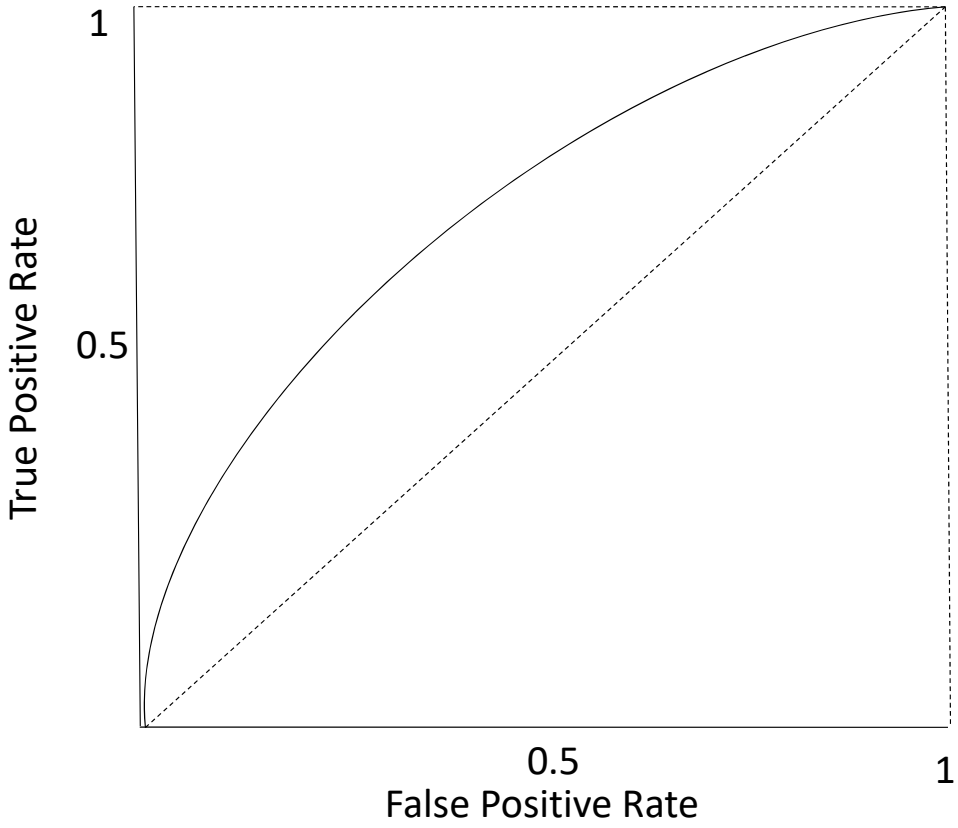
$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **F-measure**

$$\text{f1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- **Precision** TP/(TP+FP), la frazione di predizioni positive corrette
- **Recall** TP/(TP+FN), la frazione di predizioni positive corrette sul totale degli item rilevanti

Receiver Operating Characteristic (ROC) Curve



- È un metodo grafico per visualizzare la relazione fra *true positive rate* e *false positive rate*
- Il punto ottimale nel grafico è in alto a sinistra (tutte le istanze positive sono classificate correttamente, nessuna istanza negativa è classificata erroneamente come positiva)
- La diagonale rappresenta la predizione casuale delle classi
- L'area sottesa dalla curva (AUC) Rappresenta una misura della qualità dell'algoritmo

Metriche di valutazione

- **Mean Absolute Error**

$$\mathrm{MAE} = \frac{\sum_{(u,i)} |p_{u,i} - r_{u,i}|}{\mathrm{N}}$$

- **Root Mean Square Error**

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(u,i)} (p_{u,i} - r_{u,i})^2}{\mathrm{N}}}$$

dove p = prediction, r = rating, u = user, i = item, N = # item

Root Mean Square Error

Per valutare la vicinanza del prodotto a M si può usare RMSE

- Si fa la somma dei quadrati delle differenze fra gli elementi di M e i corrispondenti elementi di UV
- Si calcola la media dei quadrati dividendoli per il numero di elementi non vuoti di M
- Si calcola la radice quadrata della media

Dimensionality reduction: clustering

- Dal momento che l'utility matrix è sparsa, è difficile ottenere misure di similarità significative (sia fra utenti che fra item)
- Per due item appartenenti alla stessa categoria, ci saranno pochissimi utenti che li hanno valutati o comprati entrambi
- Anche se due utenti hanno espresso una preferenza per la stessa categoria, è molto probabile che non abbiano comprato alcun item in comune
- Si può risolvere facendo un *clustering* (raggruppamento) degli item o degli utenti
  - Si sceglie una distanza e un algoritmo di clustering e si raggruppano gli item/utenti ottenendo una matrice di dimensione più piccola
  - Una buona scelta è raggruppare gli item/utenti in un numero di cluster pari alla metà degli item/utenti
  - *Items clustering*, le colonne della matrice rappresentano i cluster degli item, l'elemento corrispondente all'utente U e al cluster C è la media dei rating che U ha dato agli item del cluster C che ha valutato. Se U non ha dato rating per nessun elemento del cluster C, allora l'elemento della matrice è vuoto
  - *Users clustering*, le righe della matrice rappresentano i cluster degli utenti, e le colonne i cluster degli item (punto precedente), l'elemento corrispondente al cluster U e al cluster C è la media dei rating che gli utenti del cluster U hanno dato agli item del cluster C che hanno valutato

Dimensionality reduction: clustering

- Il processo di clustering può essere ripetuto varie volte
  - *items* clustering, merge delle colonne
  - *users clustering*, merge delle righe
- Una volta ottenuta la matrice ridotta si possono calcolare gli elementi
- Se vogliamo predire l'elemento per l'utente U e l'item I
  - Si trovano i cluster ai quali appartengono U ed I (per esempio C e D rispettivamente)
  - Se l'elemento della matrice per C e D non è vuoto, si usa questo valore come stima del valore U-I presente nella utility matrix originaria
  - Se l'elemento della matrice per C e D è vuoto si considerano i cluster simili a C e D

Dimensionality Reduction

- Idea: per predire gli spazi vuoti della utility matrix si può scomporre (fattorizzare) la matrice originale nel prodotto di matrici più piccole
- Si utilizza un procedimento chiamato SVD (Singular Value Decomposition)

$$M = U_{m,m}\, S_{m,n} V^T_{n,n}$$

- S è una matrice diagonale

Dimensionality Reduction: UV-decomposition

- Stimare gli elementi vuoti della utility matrix scomponendola nel prodotto di due matrici
- Questa tecnica si chiama UV-decomposition, ed è un'applicazione della teoria più generale conosciuta come SVD (Single Value Decomposition)
- Supponendo di avere una matrice $\mathrm{M_{nm}}$ (n utenti, m item), vogliamo trovare una matrice $\mathrm{U_{nd}}$ e una matrice $\mathrm{V_{dm}}$ tali che UV approssima $\mathrm{M_{nm}}$ nei suoi elementi vuoti
  - UV-decomposition di M

UV-decomposition (esempio)

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

Root Mean Square Error (esempio)

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$(5-2)^2+(2-2)^2+(4-2)^2+(4-2)^2+(3-2)^2=18$
$(3-2)^2+(1-2)^2+(2-2)^2+(4-2)^2+(1-2)^2=7$
$(2-2)^2+?+(3-2)^2+(1-2)^2+(4-2)^2=6$
$(2-2)^2+(5-2)^2+(4-2)^2+(3-2)^2+(5-2)^2=23$
$(4-2)^2+(4-2)^2+(5-2)^2+(4-2)^2+?=21$

23 è il numero di elementi non vuoti di M

$$\text{RMSE} = \sqrt{\frac{18+7+6+23+21}{23}} = \sqrt{\frac{75}{23}} = 1.806$$

UV-decomposition: calcolo incrementale

- L'obbiettivo è trovare la decomposizione UV che minimizza RMSE
- Si inizia scegliendo arbitrariamente U e V, e si aggiustano ripetutamente in modo da diminuire il valore di RMSE
- Generalmente si possono fare aggiustamenti a un elemento per volta, ma anche più complicati
- Si trovano più *minimi locali*, ovvero matrici U e V tali che nessun aggiustamento dei loro elementi riduce il valore di RMSE
- Soltanto uno dei minimi locali è il *minimo globale*, ovvero le matrici U e V tali che RMSE è minimo
- Per aumentare la probabilità di trovare il minimo globale è necessario provare molte combinazioni iniziali di U e V
- Non c'è la garanzia che il miglior minimo locale trovato sia il minimo globale

UV-decomposition: calcolo incrementale (esempio)

Consideriamo l'elemento u_{11} come una variabile x

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} \Rightarrow \begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2+(4-(x+1))^2+(3-(x+1))^2$$
$$= (4-x)^2+(1-x)^2+(3-x)^2+(3-x)^2+(2-x)^2$$

Per minimizzare si prende la derivata rispetto a x e si uguaglia a 0

$$-2 \times ((4-x)+(1-x)+(3-x)+(3-x)+(2-x)) = -2 \times (13-5x) = 0 \Rightarrow x = 2.6$$

Con questo valore la somma dei quadrati degli errori sulla prima riga si riduce da 18 a 5.2, RMSE da 1.806 a 1.644

UV-decomposition: calcolo incrementale (esempio)

Consideriamo l'elemento v_{11} come una variabile y

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} \Rightarrow \begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$(5-(2.6y+1))^2+(3-(y+1))^2+(2-(y+1))^2+(2-(y+1))^2+(4-(y+1))^2$$
$$=(4-2.6y)^2+(2-y)^2+(1-y)^2+(1-y)^2+(3-y)^2$$

Per minimizzare si prende la derivata rispetto a y e si uguaglia a 0

$$-2\times\big(2.6y(4-2.6y)+(2-y)+(1-y)+(1-y)+(3-y)\big)=0 \Rightarrow y={}^{17.4}/_{10.76}=1.617$$

UV-decomposition: calcolo incrementale (esempio)

Cosa succede nei casi in cui gli elementi di M sono vuoti? Consideriamo l'elemento u_{31} come una variabile z

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} \rightarrow \begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ z & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1.617 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 5.204 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2.617 & 2 & 2 & 2 & 2 \\ 1.617z+1 & z+1 & z+1 & z+1 & z+1 \\ 2.617 & 2 & 2 & 2 & 2 \\ 2.617 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$(2-(1.617z+1))^2+(3-(z+1))^2+(1-(z+1))^2+(4-(z+1))^2$$
$$=(1-1.617z)^2+?+(2-z)^2+(-z)^2+(3-z)^2$$

Per minimizzare si prende la derivata rispetto a z e si uguaglia a 0

$$-2\times(1.617(1-1.617z)+(2-z)+(-z)+(3-z))=0 \Rightarrow z={}^{6.617}/_{5.615}=1.178$$

UV-decomposition: formula generale

- Consideriamo l'utility matrix M_{nm} con alcuni elementi vuoti, U_{nd} e V_{dm}
- $m_{ij} \in M$, $u_{ij} \in U$, $v_{ij} \in V$, $p_{ij} \in P = UV$
- Vogliamo variare u_{rs} e trovare il valore di questo elemento che minimizza RMSE fra M e UV
- $p_{rj} = \sum_{k=1}^{d} u_{rk} v_{jk} = \sum_{k \neq s} u_{rk} v_{kj} + x\, v_{sj}$, dove $x = u_{rs}$

Se m_{rj} è un valore non vuoto di M allora il contributo di questo elemento alla somma dei quadrati degli errori è

$$(m_{rj} - p_{rj})^2 = (m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj})^2$$

UV-decomposition: formula generale

La somma dei quadrati degli errori che risentono del valore $x = u_{rs}$ è

$$\sum_{j}(m_{rj} - \sum_{k \neq s} u_{rk}\, v_{kj} - x v_{sj})^2$$

Derivando la precedente rispetto a x e uguagliando a 0 si trova il valore di x che minimizza RMSE

$$\sum_{j} -2 v_{sj}(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} - x v_{sj}) = 0$$

dove $\sum_j$ indica la sommatoria su tutti gli indici j tali che m_{rj} è non vuoto

UV-decomposition: formula generale

Risolvendo per x si trova

$$x = \frac{\sum_j v_{sj}(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj})}{\sum_j v_{sj}^2}$$

Analogamente si trova il valore ottimo di V, con $y = v_{rs}$

$$y = \frac{\sum_i u_{ir}(m_{is} - \sum_{k \neq r} u_{ik} v_{ks})}{\sum_i u_{ir}^2}$$

dove $\sum_i$ indica la sommatoria su tutti gli indici i tali che m_{is} è non vuoto

UV-decomposition: algoritmo

Consiste di 4 fasi

- Preprocessing della matrice M
- Inizializzazione di U e V
- Ordinamento dell'ottimizzazione degli elementi di U e V
- Fine dell'ottimizzazione

Preprocessing della matrice M

- Si sottrae da ogni elemento non vuoto m_{ij} il rating medio per l'utente i
- Nella matrice risultante si sottrae il rating medio dell'item j
- Un'alternativa alle precedenti è di sottrarre a m_{ij} la media dei rating medi dell'utente i e l'item j
- Se si normalizza M allora è necessario denormalizzare
  - Qualunque metodo si utilizzi per stimare e di m_{ij} nella matrice normalizzata, allora il valore predetto per m_{ij} è pari a e più l'ammontare sottratto nel processo di normalizzazione

Inizializzazione

- È importante fare una scelta casuale della soluzione iniziale UV
- L'esistenza di molteplici minimi locali giustifica la scelta di eseguire più ottimizzazioni per cercare il minimo globale
- Una buona scelta per U e V consiste nel dare a ogni elemento lo stesso valore, in modo che gli elementi del prodotto UV siano la media degli elementi non vuoti di M
  - Se M è stata normalizzata, tale valore è pari a 0
- Se d è stata scelta come dimensione dei lati "corti" di U e V, e a è la media degli elementi non vuoti di M, allora gli elementi di U e V dovrebbero essere $\sqrt{a/d}$
- Se si vogliono molti punti di avvio per U e V possiamo variare il valore $\sqrt{a/d}$ casualmente e indipendentemente per ciascuno degli elementi
  - Per esempio si può aggiungere a ciascun elemento un valore casuale con distribuzione normale, media 0 e deviazione standard predefinita, oppure un valore distribuito uniformemente in $[-c, c]$ per qualche c

Ottimizzazione

- Per raggiungere un minimo locale da un dato punto di inizio di U e V si deve scegliere un ordine con il quale visitare gli elementi di U e V
  - Per esempio riga per riga, visitandoli in modalità round-robin (circolare)
- Il fatto di aver ottimizzato un elemento non significa che non si possa trovare un miglior valore per esso dopo aver ottimizzato gli altri elementi
  - Bisogna visitare gli elementi ripetutamente finché non siamo certi di essere arrivati all'ottimo
- Alternativamente si possono seguire diversi percorsi di ottimizzazione a partire da un singolo valore, scegliendo randomicamente gli elementi da ottimizzare
- Per essere certi che ogni elemento è stato considerato in ogni round possiamo invece scegliere una permutazione degli elementi e seguire quell'ordine per ogni round

Convergenza al minimo

- Teoricamente a un certo punto RMSE = 0
- Praticamente, dal momento che vi sono molti più elementi non vuoti nella matrice M che in U e V considerate insieme, non è possibile ridurre RMSE a 0
- Può non essere conveniente proseguire con i round di ottimizzazione se il miglioramente conseguito non è considerato accettabile
  - Si può impostare una threshold e fermarsi se il miglioramente è inferiore ad essa
  - Si possono considerare i miglioramenti ottenuti variando i singoli elementi di M e fermarsi quando il massimo miglioramento è sotto una soglia predefinita

Gradient Descent

- La tecnica per trovare una UV-decomposition è un esempio di *gradient descent*
- Dati alcuni punti (gli elementi non vuoti di M) per ciascun punto si trova la direzione di cambiamento che decrementa maggiormente RMSE
- Una variante consiste nel considerare randomicamente soltanto una frazione dei dati
  - *Stochastic gradient descent*
- Un'altra variante
  - ALSWR (Alternating-Least-Squares with Weighted-λ-Regularization)

Overfitting

- Può accadere che il processo di ottimizzazione porti a trovare un RMSE piccolo, ma il modello trovato non predice bene i dati
  - Si evita di favorire le prime componenti che devono essere ottimizzate cambiando il valore di un elemento soltanto di una frazione, per esempio la metà rispetto al valore da esso a quello ottimizzato
  - Si blocca il processo di visita degli elementi di U e V molto prima che il processo converga
  - Si prendono molte differenti decomposizioni UV e nel predire un nuovo elemento per la matrice M si prende la media dei risultati di ciascuna decomposizione

Altre tecniche: regole di associazione

- È una tecnica di *data mining* per trovare relazioni fra variabili
- Le associazioni sono rappresentate come regole
  - *antecedente->conseguente*
  - Se un cliente compra A, allora comprerà anche B
- La qualità di una regola di associazione è valutata sulla base di
  - *Support*
  - *Confidence*

Metriche per regole di associazione

- **Support**, rapporto fra il numero di istanze che soddisfano sia l'antecedente che il conseguente e il numero totale di istanze N

$$\mathrm{Sup}(\mathrm{R_i}) = \frac{\mathrm{n(antecedente \cdot conseguente)}}{\mathrm{N}} = \frac{\mathrm{TP}}{\mathrm{N}}$$

- **Confidence** (Precision o Positive Predictive Value, PPV), la frazione di istanze positive di una regola

$$\mathrm{Conf}(\mathrm{R_i}) = \frac{\mathrm{n(antecedente \cdot conseguente)}}{\mathrm{n(antecedente)}} = \frac{\mathrm{TP}}{\mathrm{TP + FP}}$$

- **Coverage**, la percentuale di istanze coperte da una regola

$$\mathrm{Cov}(R_i) = \frac{n(Cond)}{N}$$

Altre tecniche: Slope One

- Si calcola la media delle differenze fra utenti che hanno valutato gli stessi item che vogliamo predire
- U_1: A = 5, B = 3, differenza 2
- U_2: A = 3, B = 4, differenza -1
- Differenza media (A – B) = (2+(-1))/2=0.5
- U_3: B = 2, quindi

$$A = B + 0.5 = 2 + 0.5 = 2.5$$

Decision Trees

- Un'alternativa all'uso di *item profile+utility matrix* è la costruzione di un classificatore, per esempio un albero di decisione
- È una collezione di nodi, organizzati come un albero binario
- Approccio top-down: le foglie dell'albero corrispondono ai *like* o *dislike*
- I nodi intermedi corrispondono alle caratteristiche degli item (feature)
- Per classificare un'istanza si naviga l'albero e si seguono i percorsi corrispondenti ai valori osservati delle feature (predicato vero: sx, falso: dx)
- Raggiunta una foglia il processo termina e la classe della foglia è assegnata all'istanza (*liked*/*not liked*)
- Un esempio di algoritmo per generare un decision trere è C4.5

Naïve Bayes

- Utilizza il teorema di Bayes per predire la classe, assumendo l'indipendenza degli attributi dalla categoria di appartenenza
- Un classificatore Bayesiano assegna un insieme di attributi A_1, $A_{2, \ldots,}$ A_n a una classe C così che P(C| A_1, $A_{2, \ldots,}$ A_n) è massima

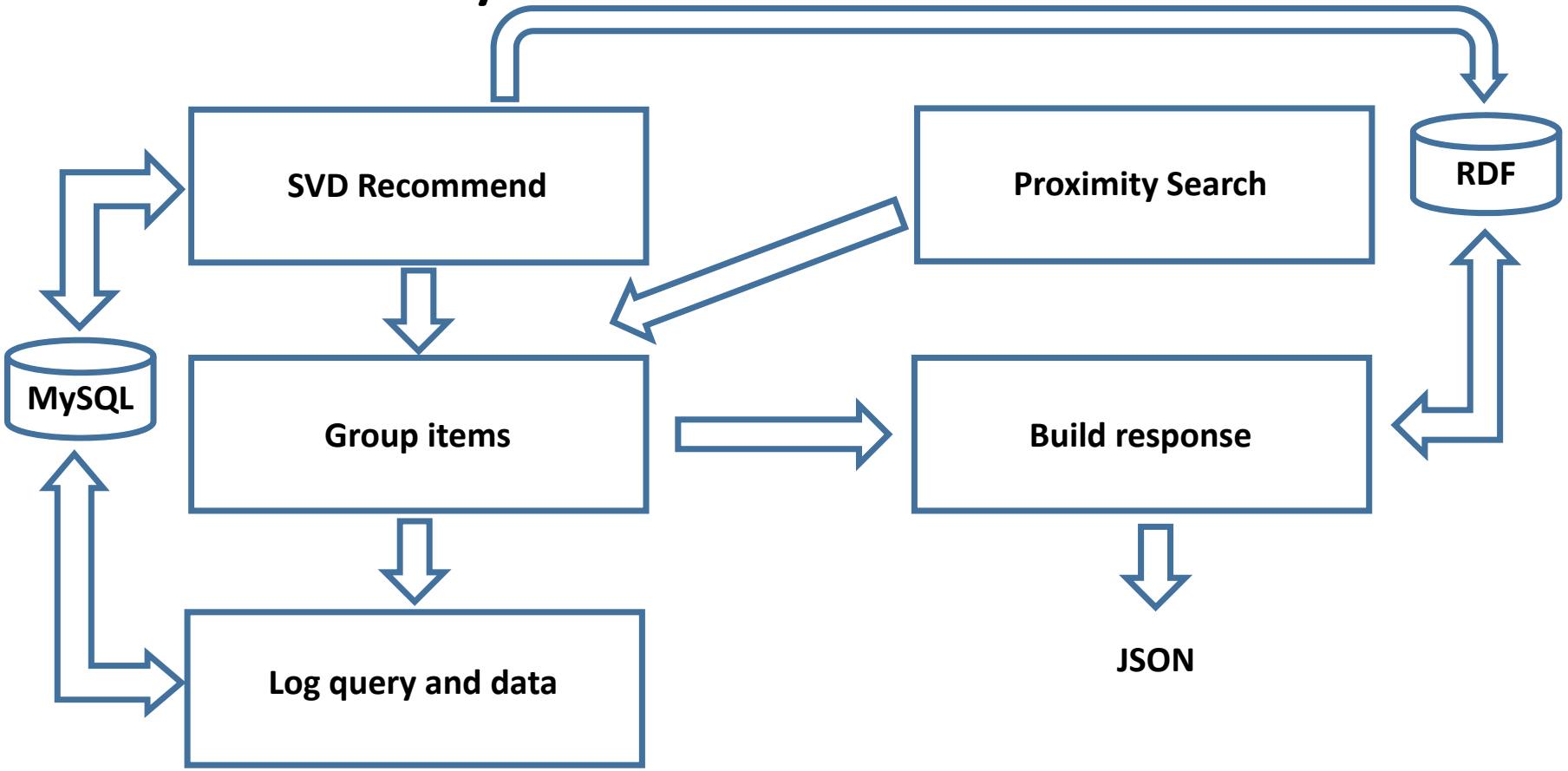
Km4City Recommender

- Implementato come una Web application utilizzata dall'app mobile *Firenze Dove Cosa* (Android, iOS, Windows Phone) che mostra tutti i servizi che sono vicini alla propria posizione e permette di navigare nella città
- È un sistema ibrido che combina i vantaggi di SVD con una ricerca per prossimità
- Vengono proposti suggerimenti suddivisi in gruppi di interesse (Istruzione, Servizi Finanziari, Salute, Hotel, Posti vicini, Servizi e Utilità, Cose da fare, Mobilità, Vino e Cibo, Bus, Eventi, Meteo, Twitter)
- I dati riguardano tutta la Toscana, con particolare attenzione alla provincia di Firenze: provengono dal MIIC della regione Toscana, dal LAMMA, dall'osservatorio Trasporti e gestore del traffico, dal Comune di Firenze, ecc.
- L'utente può esprimere un *dislike* su una categoria o sottocategoria
- Privilegia i contenuti mai suggeriti all'utente
- Tiene conto del comportamento implicito dell'utente (visualizzazione di contenuti) per calibrare il rating da attribuire a ciascun item

Km4City Recommender

- Un item appartiene a una subclass
- Una subclass appartiene a una macroclass
- Una macroclass appartiene a un gruppo
- Esempio
  - $Hotel \in http://www.disit.org/km4city/schema\#Accommodation$
- Score, si pesa l'importanza delle macroclass e subclass
  - $preference(utente) = \alpha \times SC + \beta \times MC$
  - $\alpha = 1, \beta = 0.6$
  - $(\mathrm{SC} = \text{subclass score}, \mathrm{MC} = \text{macroclass score})$
  - SC e MC equivalgono al numero di visualizzazioni di contenuti ad esse riferibili

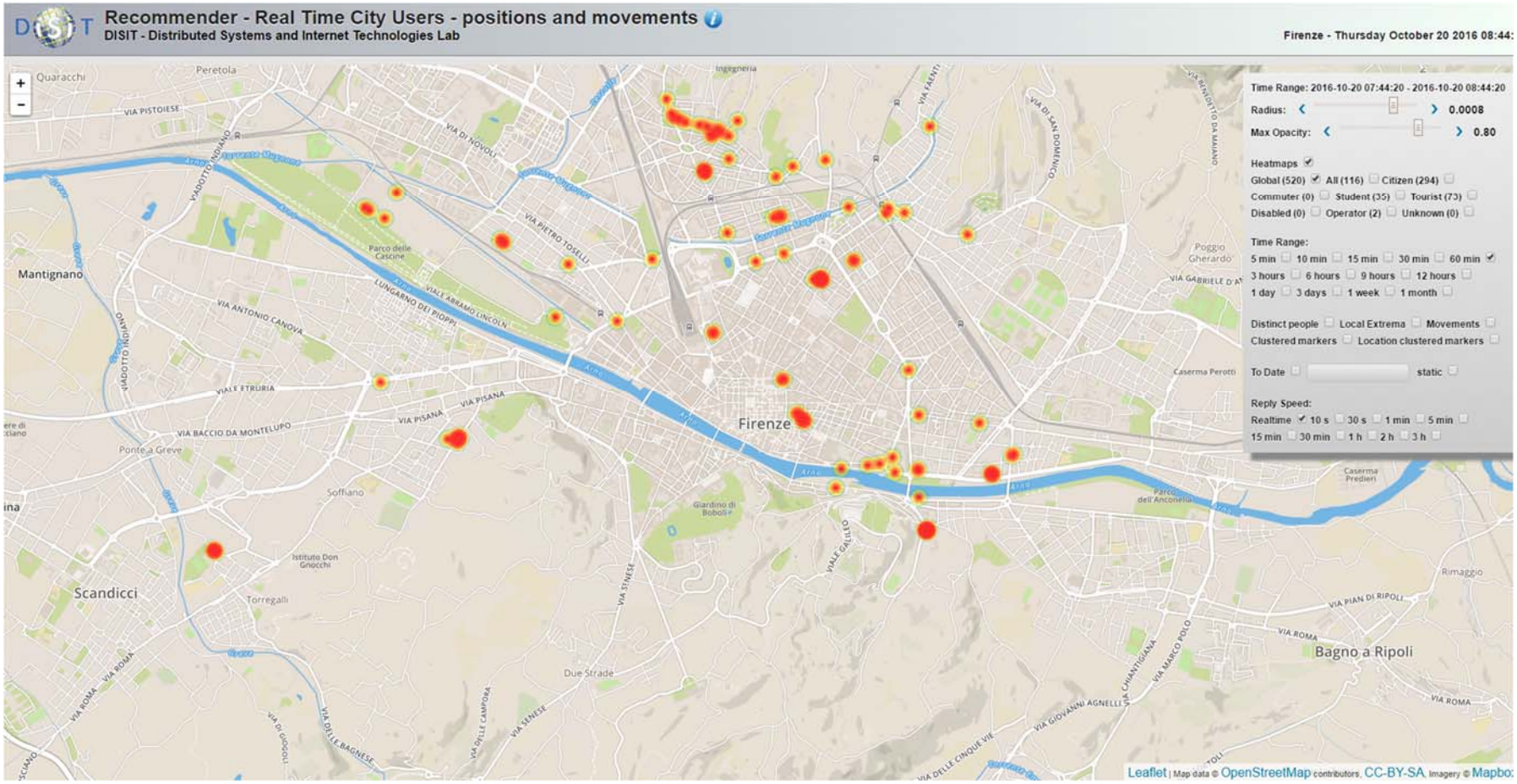
Km4City Recommender



Km4City Recommender (prestazioni)

- Work in progress: stimare precision e recall
  - 1M di record derivanti dalle azioni degli utenti
  - La maggior parte degli utenti genera pochi dati
  - I dati devono essere ripuliti in modo da considerare soltanto gli eventi generati direttamente dalle azioni degli utenti che utilizzano l'app
  - Si considerano soltanto gli utenti che interagiscono maggiormente con il sistema (i.e., che hanno visto almeno N item)
  - Possibile semplificazione: si assumono come rilevanti gli item appartenenti alla stessa macroclass o subclass di un item già valutato
    - È un'approssimazione che può introdurre molto rumore e falsare i risultati
  - Si cerca di stimare la *precision at n*
    - Per ciascun utente si determinano le top N preferenze, si rimuovono dal data model e poi si trova la percentuale di questi N item inclusi nelle top N raccomandazioni per l'utente

Km4City Recommender (movimenti realtime)



Km4City Recommender (analisi dei flussi)