UNIVERSITÀ DEGLI STUDI DI BARI

FACOLTÀ DI SCIENZE MM. FF. NN.

Corso di Laurea Specialistica in Informatica

Tesi di Laurea in
Linguaggi e Traduttori

Sviluppo di servizi per la fruizione personalizzata di beni culturali basati su un sistema di raccomandazione content-based

Relatori:
Chiar.mo prof. Giovanni SEMERARO
dott. Marco DE GEMMIS
dott. Pasquale LOPS

Laureando:
Massimo BUX

ANNO ACCADEMICO 2006-2007

Alla mia famiglia
A Simona

Indice degli argomenti

i. Introduzione

Quante volte siamo andati in un negozio di abbigliamento, indecisi su cosa acquistare, e, indicando ad una commessa i nostri gusti, questa ci consigliava i capi a noi più adatti? Allo stesso modo, per chi è più "esperto" di Informatica e di Internet esistono determinati siti che, previa compilazione di un questionario, ed indagando sugli oggetti già acquistati, indicano i prodotti più vicini ai gusti dell'utente in questione. Tecnicamente, questi prendono il nome di sistemi di raccomandazione.

Tipicamente si tratta di sistemi che confrontano l'utente – o i suoi gusti – con alcune informazioni in essi memorizzate. I dati, inoltre, vengono collezionati o tramite questionari, o tramite votazioni (classicamente in scala da 1 a 5) di alcuni oggetti in vendita, o, in alcuni casi, mostrando due oggetti e chiedendo all'utente di sceglierne il migliore.

I sistemi di raccomandazione sono diventati un'importante area di ricerca sin dalla comparsa dei primi lavori sul filtraggio collaborativo – una tecnica che raccoglie strumenti in grado di recuperare informazioni sugli interessi di un dato insieme di utenti, predicendole da una massa molto ampia di dati.

Nell'ultimo decennio è stato fatto moltissimo lavoro in quest'area, sia in ambito industriale che in ambito universitario; uno degli sforzi maggiori, infatti, è proprio quello di venire incontro al sovraccarico informativo – cioè di aiutare a districarsi nell'enorme quantità di informazioni presenti, sia nella vita reale che sul Web.

Tutti questi concetti potrebbero essere estesi in un ambito molto diverso dalla semplice vendita di prodotti: la fruizione di beni culturali in un museo. È proprio questo l'obiettivo del mio lavoro di tesi, un progetto che

nasce dall'esigenza non solo di creare, ma di migliorare, tramite raccomandazioni – che sfruttano anche concetti collaborativi tipici del Web 2.0 – dei servizi ideati in un progetto il cui obiettivo è la costruzione di una piattaforma per la creazione di servizi multimodali mobili. Questo progetto prende il nome di CHAT – Cultural Heritage fruition & e-learning applications of new Advanced (multimodal) Technologies.

Il *capitolo 1*, dunque, offrirà una panoramica sullo stato dell'arte degli attuali sistemi di raccomandazione. Dunque, si spazierà fra i più noti progetti di ricerca, andando a marcare quelle che sono le differenze fra i sistemi *content-based*, cioè basati sul contenuto degli item da raccomandare, quelli *collaborativi*, che sfruttano cioè aspetti quali i voti degli altri utenti per migliorare le raccomandazioni stesse, e quelli *ibridi*, che combinano aspetti dei due precedenti.

Il *capitolo 2* si concentrerà sul primo aspetto importante del lavoro di tesi: ITem Recommender. Questo è un progetto di ricerca dell'Università degli Studi di Bari, un recommender, dunque, che lavorava su item statici, tipicamente film, divisi in slot, e, tramite classificatore bayesiano e due classi di appartenenza, aveva il compito di decidere se un item era di gradimento o meno per un utente, dati un insieme di voti. Il recommender è stato completamente reingegnerizzato, suddiviso in packages, svincolato dalla staticità degli slot, parametrizzato.

Il *capitolo 3* si occuperà di presentare il progetto CHAT sotto tutti i suoi punti di vista – nello specifico, una panoramica sui suoi obiettivi e una visione dell'architettura del sistema. Dopo questa prima fase, si individueranno gli scenari in cui potrebbe essere possibile integrare ITR, mostrando dunque in che maniera portare a termine questo task.

Il *capitolo 4* è dedicato proprio alla progettazione ed allo sviluppo dei servizi, previsti in CHAT, e adattati ad un museo, la *Pinacoteca dei Musei Vaticani*, scelto ad hoc per gli stessi. In primo luogo, verrà illustrato il motivo della scelta della Pinacoteca e in che maniera i suoi contenuti sono stati estratti dal sito ufficiale. Secondariamente, verranno mostrati i servizi, realizzati tramite un Web Services e che si adattano, tramite le raccomandazioni fornite da ITR, al singolo utente.

Il *capitolo 5* sarà, invece, completamente incentrato sulle sperimentazioni effettuate sui servizi implementati e descritti al capitolo precedente. Difatti, lo scopo della sperimentazione – suddivisa in tre passi: *preparazione*, *esecuzione* e *valutazione*, largamente esplicati nei vari paragrafi – non è quello di verificare la bontà specifica di ITR, bensì quello di comprendere se l'*utilizzo* di ITR all'interno di CHAT porti a un miglioramento sostanziale dei servizi offerti.

I *capitoli 6*, *7* ed *8*, infine, sono dedicati, rispettivamente, alle conclusioni ed agli sviluppi futuri di questo lavoro, ai ringraziamenti ed alla bibliografia utilizzata per la stesura della tesi.

1. Stato dell’arte dei sistemi di raccomandazione

Come già detto in fase introduttiva, c’è una grande abbondanza di applicazioni che aiutano gli utenti nel far fronte all’information overload e a fornire raccomandazioni personalizzate. Esempi di queste applicazioni includono raccomandazioni di libri, CD ed altri prodotti su Amazon.com, film su MovieLens, e notizie su VERSIFI Technologies (precedentemente AdaptiveInfo.com). Inoltre, alcuni di questi hanno incorporato caratteristiche di raccomandazione nei loro server.

Comunque, a dispetto di tutti questi notevoli avanzamenti, l’attuale generazione di sistemi di raccomandazione richiede ancora ulteriori miglioramenti per rendere i metodi di raccomandazione più efficaci ed applicabili ad un range ancora più ampio di applicazioni della vita reale, per esempio raccomandazioni di vacanze, alcuni tipi di servizi finanziari per gli investitori, e prodotti per acquistare in un negozio tramite una “shopping cart intelligente”.

Questi miglioramenti includono diversi metodi per rappresentare il comportamento dell’utente e le informazioni sugli item da raccomandare, metodi di modellazione delle raccomandazioni più avanzati, incorporamento di diverse informazioni di contesto nel processo di raccomandazione, utilizzo di rating multi-criterio, sviluppo di metodi di raccomandazione meno intrusivi e più flessibili che si relazionano anche alle misure che determinano più efficacemente la performance dei sistemi di raccomandazione.

In questo capitolo verranno descritti diversi modi di estendere le capabilities dei sistemi di raccomandazione. Comunque, prima di fare questo, verrà prima presentata un’indagine esplorativa dello stato dell’arte nei sistemi di raccomandazione. Inoltre, verranno identificate diverse

limitazioni nell'attuale generazione dei metodi di raccomandazione e discussi alcuni approcci iniziali per estendere le loro capabilities.

1.1 Un'indagine sui sistemi di raccomandazione

Sebbene le radici dei recommender systems possano essere rintracciate, andando indietro nel tempo, nei lavori sulle scienze cognitive, sulle teorie di approssimazione, sull'information retrieval, sulle teorie di previsione, ed hanno anche collegamenti alla management science ed alla modellazione delle scelte del consumatore nel marketing, i sistemi di raccomandazione sono emersi come un'area di ricerca indipendente alla metà degli anni '90, quando i ricercatori hanno iniziato a focalizzare le loro attenzioni sui problemi di raccomandazione che si relazionano *esplicitamente* alle strutture di voto.

Nella sua formulazione più comune, il problema della raccomandazione è ridotto al problema di stimare i voti per gli item che non sono stati visti da un utente. Intuitivamente, questa stima è solitamente basata sui voti dati da questo utente ad altri item e su alcune altre informazioni che saranno formalmente descritte dopo. Una volta che si possono stimare i voti per gli item non ancora votati, si possono raccomandare agli utenti l'item/gli items con i più alti voti appena stimati.

Più formalmente, il problema della raccomandazione può essere formulato come segue: sia C l'insieme di tutti gli utenti e sia S l'insieme di tutti i possibili item che possono essere raccomandati, come ad esempio libri, film o ristoranti.

Lo spazio S dei possibili item può essere molto ampio, spaziando dalle centinaia, alle migliaia, o addirittura ai milioni di item in alcune

applicazioni, come la raccomandazione di libri o CD. In maniera similare, lo spazio utente può essere molto ampio – milioni in alcuni casi.

Sia *u* una funzione di utility che misura l'utilità dell'item *s* per l'utente *c*, per esempio, $u : C \times S \to R$, dove *R* è un insieme totalmente ordinato (per esempio, interi non negativi o numeri reali all'interno di un certo range). Dunque, per ogni utente $c \in C$ vogliamo scegliere quale item $s' \in S$ massimizza l'utility dell'utente. Più formalmente:

$$\forall c \in C, \quad s'_c = \arg\max_{s \in S} u(c, s) \qquad (1)$$

Nei sistemi di raccomandazione, l'utilità di un item è solitamente rappresentata da un voto *(rating)*, che indica quanto sia piaciuto un particolare item ad un particolare utente, ad esempio Simona Lisena ha dato al film "La casa sul lago del tempo" il voto 8 (su 10). Comunque, come indicato in precedenza, in generale l'utilità può essere una funzione arbitraria, e può includere una funzione di profitto.

A seconda dell'applicazione, l'utility *u* può essere o specificata dall'utente, come è spesso fatto per i voti *user-defined*, o calcolata dall'applicazione, come nel caso delle funzioni di utility *profit-based*.

Ogni elemento dello spazio utente *C* può essere definito con un *profilo* che include varie caratteristiche dell'utente, come età, sesso, stipendio, stato civile, ecc. Nel caso più semplice, il profilo può contenere solo un unico elemento, come l'ID utente. In maniera similare, ogni elemento dello spazio degli item *S* è definito da un insieme di caratteristiche. Per esempio, in un'applicazione per la raccomandazione di film, dove *S* è una collezione di film, ognuno di essi può essere rappresentato non soltanto

dal suo ID, ma anche dal suo titolo, genere, regista, anno di produzione, attori protagonisti, ecc.

Il problema centrale dei sistemi di raccomandazione risiede nel fatto che l'utility *u* solitamente non è definita nell'intero spazio $C \times S$, ma solo in alcuni sottoinsiemi di esso. Ciò significa che *u* ha bisogno di essere *estrapolato* per l'intero spazio $C \times S$.

Nei sistemi di raccomandazione, l'utility è tipicamente rappresentata dai voti ed è inizialmente definita solo sugli item precedentemente votati dagli utenti. Per esempio, in un'applicazione per la raccomandazione di film (come MovieLens.org), gli utenti inizialmente votano alcuni sottoinsiemi di film che hanno già visto. Un esempio di una matrice user-item rating è presentata in Tabella 1

	Matrix	Law & Order	Ghost	The Godfather
Mario	4	3	2	4
Francesca	∅	4	5	5
Marco	2	2	4	∅
Simona	3	∅	5	2

Tabella 1 - Un frammento di Matrice di Voti per un Sistema di raccomandazione di Film

In questa tabella, i voti sono specificati in una scala da 1 a 5. Il simbolo "∅" per alcuni voti in tabella indica che l'utente non ha votato il film corrispondente. Pertanto, il motore di raccomandazione dovrebbe essere in grado di stimare (predire) i voti dei film non votati e costruire raccomandazioni appropriate basate su queste predizioni.

Le estrapolazioni di voti sconosciuti da voti conosciuti sono solitamente effettuate in due passi:

1. Specificando *euristiche* che definiscono la funzione di utility e validano empiricamente la sua performance;

2. *Stimando* la funzione di utility che ottimizza alcuni criteri di performance, come l'errore quadratico medio.

Una volta che i voti sconosciuti sono stimati, le raccomandazioni effettive di un item ad un utente sono fatte selezionando il più alto rating fra tutti i voti stimati per quell'utente, in accordo alla formula (1). In alternativa, si possono raccomandare gli *N* item migliori per un utente o un insieme di utenti ad un item.

I nuovi voti degli item non ancora votati possono essere stimati in molti modi differenti, utilizzando metodi che spaziano tra machine learning, teoria dell'approssimazione, e varie euristiche. I sistemi di raccomandazione sono solitamente classificati in accordo al loro approccio alla stima dei voti e, nel prossimo paragrafo, verrà presentata una classificazione largamente accettata in letteratura e verrà dunque visionata un'indagine dei differenti tipi di sistemi di raccomandazione.

Sistemi di raccomandazione che, solitamente, sono classificati nelle seguenti categorie, basate sul modo in cui le raccomandazioni sono fatte:

- *Raccomandazioni content-based*: all'utente verranno raccomandati item simili a quelli che l'utente ha preferito in passato;
- *Raccomandazioni collaborative*: all'utente saranno raccomandati item che la gente con gli stessi gusti e preferenze hanno preferito in passato;
- *Approcci ibridi*: questi metodi combinano metodi collaborativi e content-based.

In aggiunta ai sistemi di raccomandazione che predicono il valore *assoluto* dei voti che gli utenti vorrebbero dare agli item non ancora votati (come discusso in precedenza), è stato fatto molto lavoro sul *preference-based filtering*, per predire le preferenze *relative* degli utenti.

Per esempio, in un'applicazione per la raccomandazione di film, le tecniche di preference-based filtering dovrebbero focalizzarsi nel predire il corretto *ordine relativo* dei film, piuttosto che i loro voti relativi. Comunque, in questo capitolo ci si focalizzerà principalmente sui recommenders *rating-based* poiché costituiscono il più popolare approccio ai sistemi di raccomandazione.

1.2 Metodi Content-Based

Nei metodi di raccomandazione content-based, la utility $u(c,s)$ dell'item s per l'utente c è stimata basandosi sulle utilities $u(c,s_i)$ assegnate dall'utente c agli item $s_i \in S$ che sono "simili" all'item s.

Per esempio, in un'applicazione di raccomandazione di film, per raccomandare dei film all'utente *c*, il sistema di raccomandazione content-based cerca di comprendere le cose in comune tra i film che l'utente *c* ha votato in maniera più alta nel passato (attori specifici, registi, generi, soggetto del film, ecc.). Infine, solo i film che hanno il più alto grado di similarità con le preferenze dell'utente saranno raccomandati.

L'approccio content-based alla raccomandazione ha le sue radici negli ambiti di ricerca sull'information retrieval e sull'information filtering. A causa dei rapidi e significativi avanzamenti fatti dalle communities sull'information retrieval e filtering ed a causa dell'importanza di diverse applicazioni text-based, molti sistemi attuali content-based si focalizzano sulla raccomandazione di item che contengono informazioni testuali, come documenti, siti Web, e news sites.

I miglioramenti agli approcci tradizionali di information retrieval vengono dall'uso di *profili utente* che contengono informazioni sui gusti, le preferenze ed i bisogni dell'utente. Le informazioni di profilazione possono essere elicitate dagli utenti esplicitamente, per esempio attraverso questionari, od appresi implicitamente dal loro comportamento nel tempo.

Più formalmente, sia *Content(s)* un *profilo utente*, per esempio un insieme di attributi che caratterizzano l'item *s*. Esso è solitamente calcolato estraendo un insieme di *features* dall'item *s* (il suo contenuto) ed è utilizzato per determinare l'appropriatezza dell'item per scopi di raccomandazione.

Poiché, come menzionato in precedenza, i sistemi content-based sono progettati principalmente per raccomandare item *text-based*, il contenuto in questi sistemi è solitamente descritto con *parole chiave (keywords)*. Per esempio, una componente content-based del sistema Fab [01], che

raccomanda pagine Web agli utenti, rappresenta il contenuto delle pagine Web con le sue 100 parole più importanti.

In maniera molto simile, il sistema Syskill & Webert [02] rappresenta i documenti con le 128 parole maggiormente informative. La "importanza" (detta *informativeness*) della parola k_j nel documento d_j è determinata con alcune misure di *weighting* w_{ij} che possono essere definite in molte maniere differenti.

Una delle misure maggiormente conosciute per specificare i pesi delle keyword nell'Information Retrieval è la misura *term frequency/inverse document frequency* (TF-IDF) [03] che è definita come segue: assumiamo che *N* sia il numero totale di documenti che possono essere raccomandati agli utenti e che la parola k_j appare in n_i di essi. Inoltre, assumiamo che f_{ij} sia il numero di volte che la parola k_i appare nel documento d_j. Infine, TF_{ij}, la frequenza (o la frequenza normalizzata) della parola k_i nel documento d_j è definita come:

$$TF_{ij} = \frac{f_{ij}}{\max_z f_{zj}} \qquad (2)$$

Dove il massimo è calcolato sulle frequenze f_{zj} di tutte le keyword k_z che appaiono nel documento d_j. Comunque, le keyword che appaiono in molti documenti sono inutili per distinguere fra un documento rilevante ed un documento non rilevante. Per questo motivo, la misura di inverse document frequency (IDF_i) è spesso usata in combinazione con la semplice term frequency (TF_{ij}). La inverse document frequency per la keyword k_i è solitamente definita come:

$$IDF_i = \log \frac{N}{n_i} \qquad (3)$$

Infine, il peso TF-IDF per la keyword k_i nel documento d_j è definita come:

$$w_{ij} = TF_{ij} \times IDF_i \qquad (4)$$

Ed il contenuto del documento d_j è defnito come:

$$Content(d_j) = (w_{1j}, ..., w_{kj})$$

Come scritto in precedenza, i sistemi content based raccomandano gli item simili a quelli che l'utente ha gradito in passato. In particolare, i vari item candidati sono confrontati con gli item precedentemente votati dall'utente ed gli item che maggiormente corrispondono sono raccomandati.

Più formalmente, sia *ContentBasedProfile(c)* il profilo dell'utente *c* che contiene gusti e preferenze di questo utente. Questi profili sono ottenuti analizzando il contenuto degli item precedentemente visti e votati dall'utente e sono solitamente costuiti utilizzando le tecniche di analisi delle keywords derivanti dall'information retrieval.

Per esempio, *ContentBasedProfile(c)* può essere definito come un vettore di pesi $(w_{c1}, ..., w_{ck})$, dove ogni singolo peso w_{ci} denota l'importanza della

keyword k_i per l'utente *c* e può essere calcolata dai vettori individuali dei contenuti votati utilizzando alcune tecniche.

Per esempio, alcuni approcci basati su media, come l'algoritmo di Rocchio [04], possono essere utilizzati per calcolare *ContentBasedProfile(c)* come un "vettore medio" ottenuto da vettori individuali.

Altre tecniche [02], invece, usano un classificatore Bayesiano per stimare la probabilità che un item sia gradito o meno. È stato dimostrato anche che l'algoritmo di Winnow lavora bene per questo scopo, specialmente nelle situazioni in cui ci sono molte possibili features.

Nei sistemi content-based, la funzione di utility $u(c,s)$ è solitamente definita come:

$$u(c,s) = score(ContentBasedProfile(c), Content(s)) \quad (5)$$

Utilizzando le informazioni menzionate in precedenza e riguardanti il paradigma basato su ritrovamento nelle pagine Web, sia *ContentBasedProfile(c)* dell'utente *c* che *Content(s)* del documento *s* possono essere rappresentati come vettori TF-IDF $\vec{w}_c$ e $\vec{w}_s$ di pesi delle keyword. Inoltre, la funzione di utility $u(c,s)$ è solitamente rappresentata nella letteratura dell'information retrieval da alcune euristiche di scoring definite in termini di vettori $\vec{w}_c$ e $\vec{w}_s$, come la misura di similarità del coseno [05] [06]:

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\| \vec{w}_c \|_2 \times \| \vec{w}_s \|_2} =$$

$$= \frac{\sum_{i=1}^{k} w_{ic} \cdot w_{is}}{\sqrt{\sum_{i=1}^{k} w_{ic}^2} \cdot \sqrt{\sum_{i=1}^{k} w_{is}^2}} \qquad (6)$$

Dove *K* è il numero totale di parole chiave nel sistema.

Per esempio, se l'utente *c* legge molti articoli online sull'argomento "bioinformatica", le tecniche di raccomandazione content-based saranno capaci di raccomandare altri articoli di bioinformatica all'utente *c*. Questo accadrà nel caso in cui questi articoli abbiano alcuni termini collegati alla bioinformatica (per esempio "genoma", "sequenze", "proteine") piuttosto che articoli su altri argomenti, e, perciò, *ContentBasedProfile(c)*, come definito dai vettori $\vec{w}_c$, rappresenterà quei termini k_i coi più alti pesi w_{ic}.

Di conseguenza, un sistema di raccomandazione che utilizza il coseno od altre misure di similarità collegate assegnerà una più alta utility $u(c,s)$ a quegli articoli *s* che hanno termini di bioinformatica con peso alto nel vettore $\vec{w}_s$ e bassa utility a quelli dove i termini di bioinformatica sono pesati minormente.

Accanto alle tradizionali euristiche che sono basate maggiormente sui metodi di information retrieval, sono state utilizzate altre tecniche per la raccomandazione content-based, come i classificatori Bayesiani [02] [07] e svariate tecniche di machine learning, che includono clustering, alberi decisionali, e reti neurali artificiali.

Queste tecniche differiscono dagli approcci basati sull'information retrieval poiché essi calcolano le predizioni sull'utility basandosi non su formule euristiche, come la misura della similarità del coseno, ma piuttosto sono basati su un *modello* appreso dai dati sottostanti utilizzando apprendimento statistico e tecniche di machine learning.

Per esempio, alcuni sistemi [02], basandosi su un insieme di pagine Web che sono state votate come "rilevanti" od "irrilevanti" dall'utente, utilizzano il classificatore naive Bayesiano per classificare le pagine Web non votate. Più specificatamente, il classificatore naive Bayesiano è utilizzato per stimare la probabilità a posteriori che la pagina p_j appartenga ad una certa classe C_i (ad esempio, rilevante od irrilevante), dato il set di keywords $k_{1j},...,k_{nj}$ su quella pagina:

$$P(C_i \mid k_{1j} \& ... \& k_{nj}) \qquad (7)$$

Inoltre, si assume che le keywords sono indipendenti e, perciò, la probabilità a priori è proporzionale a:

$$P(C_i)\prod_x P(k_{xj} \mid C_i) \qquad (8)$$

Mentre l'assunzione di indipendenza dalle keyword non si applica necessariamente a molte applicazioni, risultati sperimentali dimostrano che i classificatori naive Bayesiani producono ancora un'alta accuratezza di classificazione [02]. Inoltre, sia $P(k_{xj} \mid C_i)$ che $P(C_i)$ possono essere stimati dai dati di training sottostanti. Perciò, per ogni pagina p_j, la

probabilità $P(C_i \mid k_{1j} \& ... \& k_{nj})$ è calcolata per ogni classe C_i e la pagina p_j è assegnata alla classe C_i con la più alta probabilità [02].

Pur non trattando esplicitamente di fornire raccomandazioni, la community di *text retrieval* ha contribuito con diverse tecniche che sono state utilizzate nei sistemi di raccomandazione content-based.

Un esempio di queste tecniche potrebbe essere la ricerca sull'*adaptive filtering* [08] [09], che si focalizza sulla ricerca di maggiore accuratezza e ha lo scopo di identificare documenti rilevanti in maniera incrementale osservando i documenti uno per uno in un flusso continuo di documenti. Un altro esempio potrebbe essere il lavoro sul *threshold setting* [10] [11], che si focalizza nel calcolare la misura in cui i documenti debbano corrispondere ad una query data per essere rilevanti per l'utente.

1.3 Limitazioni dei metodi Content-Based

Come intuibile, i sistemi di raccomandazione content-based hanno diverse limitazioni che verranno descritte nei prossimi sottoparagrafi.

1.3.1 Limited Content Analysis

Le tecniche content-based sono limitate dalle features che sono esplicitamente associate con gli oggetti che questi sistemi raccomandano. Perciò, per far sì che si abbia un sufficiente insieme di features, il contenuto deve essere o in una forma che possa tranquillamente essere parserizzata da un computer (per esempio, in forma testuale), oppure le features devono essere assegnate agli item manualmente.

Mentre le tecniche di information retrieval lavorano meglio nell'estrazione di features da documenti testuali, alcuni altri domini hanno un problema inerente all'estrazione automatica di features. Per esempio, i metodi di estrazione automatica delle features sono difficilmente applicabili a dati multimediali, per esempio ad immagini, flussi audio e video. Inoltre, non è molto pratico assegnare attributi a mano a causa delle limitazioni delle risorse.

Un altro problema con la limited content analysis è che, se due differenti item sono rappresentati dallo stesso insieme di features, essi sono indistinguibili. Perciò, poiché i documenti basati su testo sono solitamente rappresentati dalle loro più importanti parole chiave, i sistemi content-based non possono distinguere tra un articolo scritto bene ed uno scritto male, se capita che utilizzino gli stessi termini.

1.3.2 Overspecialization

Quando il sistema può *solo* raccomandare item che totalizzano un voto alto, dato un profilo, l'utente subisce la limitazione che il sistema gli raccomanderà item che sono simili soltanto a quelli già votati.

Per esempio, una persona con nessuna esperienza nella cucina Greca non vorrebbe mai ricevere una raccomandazione persino del miglior ristorante Greco della città. Questo problema, che è stato anche studiato in altri domini, è spesso indirizzato introducendo la cosiddetta *randomness*. Per esempio, è stato proposto l'utilizzo di algoritmi genetici come possibile soluzione nel contesto dell'information filtering.

In aggiunta, il problema della overspecialization non è soltanto quello che i sistemi content-based non possono raccomandare item che sono differenti da qualsiasi cosa che l'utente abbia visto prima. In certi casi, gli

item non dovrebbero essere raccomandati se sono *troppo simili* a qualcosa che l'utente ha già visto, come per esempio differenti articoli di giornale che descrivono lo stesso evento.

Perciò, alcuni sistemi di raccomandazione content-based, come Daily Learner [12], filtrano gli item non solo se sono molto differenti dalle preferenze dell'utente, ma anche se sono troppo simili a qualcosa che l'utente ha visto in precedenza. Riassumendo, la *diversità* di raccomandazioni è spesso una feature ricercata nei sistemi di raccomandazione. Idealmente, all'utente dovrebbe essere presentato un range di opzioni e non un omogeneo insieme di alternative. Per esempio, non è necessariamente una buona idea raccomandare tutti i film di Woody Allen ad un utente a cui è piaciuto solo un suo film.

1.3.3 Problema del Nuovo Utente

L'utente deve votare un numero sufficiente di item prima che un sistema di raccomandazione content-based possa veramente comprendere le preferenze dell'utente e presentare all'utente raccomandazioni accettabili. Perciò, un nuovo utente, che ha molti pochi voti, potrebbe non ottenere raccomandazioni accurate.

1.4 Metodi Collaborativi

A differenza dei metodi di raccomandazione content-based, i sistemi di raccomandazione *collaborativi* (o *sistemi di filtering collaborativo*) cercano di predire l'utilità degli item per un particolare utente basandosi sugli item precedentemente votati dagli *altri utenti*.

Più formalmente, la utility $u(c,s)$ dell'item *s* per l'utente *c* è stimata basandosi sull'utility $u(c_j,s)$ assegnata all'item *s* per quegli utenti $c_j \in C$ che sono "simili" all'utente *c*.

Per esempio, in un'applicazione di raccomandazione di film, per raccomandare film all'utente *c*, il sistema di raccomandazione collaborativo cerca di trovare i *"pari"* dell'utente *c*, vale a dire quegli utenti che hanno gusti simili in materia di cinema (votano gli stessi film in maniera similare). Dunque, solo i film che sono maggiormente apprezzati dai "pari" dell'utente *c* saranno raccomandati.

In ambito universitario ed industriale sono stati sviluppati molti sistemi collaborativi. Si può sostenere che il sistema Grundy [13] è stato il primo sistema di raccomandazione che propose di utilizzare gli *stereotipi* come meccanismo per costruire modelli utente basati su un ammontare limitato di informazioni per ogni utente individuale. Utilizzando gli stereotipi, il sistema Grundy costruirebbe modelli utente individuali utilizzandoli per raccomandare libri rilevanti per ogni utente.

Successivamente, il sistema Tapestry si basò su ogni utente per identificare manualmente utenti con gli stessi interessi [14]. Altri sistemi come GroupLens [15] furono i primi ad utilizzare il collaborative filtering per *automatizzare* la previsione. Un altro esempio di sistema di raccomandazione collaborativa è il sistema di raccomandazione di libri di Amazon.com.

In accordo ad alcuni studi [16], gli algoritmi per la raccomandazione collaborativa si suddividono in due classi generali: *memory-based* (detti anche *heuristic-based*) e *model-based*.

Gli algoritmi *memory-based* sono essenzialmente euristiche che forniscono previsioni basate sull'intera collezione di item

precedentemente votati dagli utenti. Questo significa che il valore del rating sconosciuto r_{cs} per l'utente *c* e l'item *s* è solitamente calcolato come un'aggregazione dei voti di qualche altro (solitamente, gli *N* più simili) utente per lo stesso item *s*:

$$r_{cs} = \underset{c' \in \hat{C}}{aggr}\ r_{c's} \qquad (9)$$

Dove $\hat{C}$ denota l'insieme degli *N* utenti che sono i più simili all'utente *c* e che hanno votato l'item *s* (*N* può variare in ogni range, da 1 all'intero insieme degli utenti). Alcuni esempi delle funzioni di aggregazione sono:

$$\begin{aligned} &(a)\ \ r_{cs} = \frac{1}{N}\sum_{c' \in \hat{C}} r_{c's} \\ &(b)\ \ r_{cs} = k\sum_{c' \in \hat{C}} sim(c,c') \times r_{c's} \\ &(c)\ \ r_{cs} = \bar{r}_c + k\sum_{c' \in \hat{C}} sim(c,c') \times (r_{c's} - \bar{r}_{c'}) \end{aligned} \qquad (10)$$

Dove il moltiplicatore *k* funge da fattore di normalizzazione ed è solitamente selezionato come:

$$k = \frac{1}{\sum_{c' \in \hat{C}} | sim(c,c') |}$$

E dove il voto medio dell'utente *c*, $\bar{r}_c$, nella formula *(10c)* è definito come:

$$\bar{r}_c = \frac{1}{|S_c|} \cdot \sum_{s \in S_c} r_{cs}, \ dove \ S_c = \{s \in S \mid r_{cs} \neq \varnothing\} \qquad (11)$$

Nel caso più semplice, l'aggregazione può essere una semplice media, come definito in *(10a)*. Comunque, il più comune approccio di aggregazione è quello di usare le somme pesate, come mostrato in *(10b)*.

La misura di similarità fra l'utente *c* e l'utente *c'*, $sim(c,c')$, è essenzialmente una misura della distanza ed è utilizzata come peso, per esempio, più simili sono gli utenti *c* e *c'*, più i pesi dei voti $r_{c's}$ porteranno con sé la previsione di r_{cs}.

È importante notare come $sim(x,y)$ è un'euristica introdotta per essere capaci di far differenza tra livelli di similarità utente (per esempio, per poter trovare un insieme di "utenti più vicini" o "nearest neighbor" per ogni utente) e, allo stesso tempo, semplificare la procedura di stima dei voti.

Come mostrato in *(10b)*, diverse applicazioni di raccomandazione possono utilizzare le proprie misure di similarità dato che i calcoli sono normalizzati utilizzando il fattore di normalizzazione *k*, come detto in precedenza. Le due misure di similarità più comunemente usate verranno descritte sotto.

Un problema che si ha con l'utilizzo delle somme pesate, come in *(10b)*, è che non si tiene conto del fatto che diversi utenti potrebbero utilizzare la scala dei voti in maniera diversa. La somma pesata *aggiustata*, mostrata in *(10c)*, è stata largamente utilizzata per migliorare questa limitazione. In

questo approccio, invece di utilizzare i valori assoluti dei voti, le somme pesate usano le loro deviazioni dal voto medio dell'utente corrispondente.

Un altro modo di superare gli usi differenti della scala di voto è di sviluppare un filtraggio *preference-based*, che si focalizza sulla previsione delle preferenze *relative* degli utenti invece dei valori di rating assoluti.

Diversi approcci sono stati utilizzati per calcolare la similarità $sim(c,c')$ tra utenti nei sistemi di raccomandazione collaborativi. Nella maggior parte di questi approcci, la similarità tra due utenti è basata sui voti agli item che *entrambi* hanno votato.

I due approcci più popolari sono *correlazione* e *cosine-based*. Per presentare questi due sistemi, assumiamo che S_{xy} sia l'insieme di tutti gli item votati da entrambi gli utenti *x* ed *y*, cioè $S_{xy} = \{s \in S \mid r_{xs} \neq \varnothing \ \& \ r_{ys} \neq \varnothing\}$.

Nei sistemi di raccomandazione collaborativi, S_{xy} è usato principalmente come risultato intermedio per calcolare i "nearest neighbors" dell'utente *x* ed è spesso calcolato in maniera semplice, per esempio calcolando l'intersezione degli insiemi S_x ed S_y. Dunque, nell'approccio *correlation-based*, viene utilizzato il coefficiente di correlazione di Pearson per calcolare la similarità:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x) \cdot (r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2 \cdot \sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}} \qquad (12)$$

Nell'approccio *cosine-based*, i due utenti *x* ed *y* sono trattati come due vettori in uno spazio *m*-dimensionale, dove $m = |S_{xy}|$. Infine, la similarità

tra due vettori può essere misurata calcolando il coseno dell'angolo fra di essi:

$$sim(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \|_2 \times \| \vec{y} \|_2} =$$

$$= \frac{\sum_{s \in S_{xy}} r_{xs} \cdot r_{ys}}{\sqrt{\sum_{s \in S_{xy}} r_{xs}^2} \cdot \sqrt{\sum_{s \in S_{xy}} r_{ys}^2}} \qquad (13)$$

Dove $\vec{x} \cdot \vec{y}$ denota il prodotto scalare tra i vettori $\vec{x}$ ed $\vec{y}$.

È importante notare come differenti sistemi di raccomandazione possono utilizzare differenti approcci per implementare il calcolo di similarità tra utenti e le stime dei voti in maniera quanto più efficiente possibile.

Una strategia comune è di calcolare *tutte* le similarità tra utenti $sim(x, y)$ (incluso il calcolo di S_{xy}) in anticipo e ricalcolandole solo una volta ogni tanto (dato che la rete di utenti simili tra di loro solitamente non cambia drasticamente in un breve periodo). Poi, ogni qualvolta l'utente richieda una raccomandazione, i voti possono essere efficientemente calcolati su richiesta utilizzando similarità precalcolate.

Inoltre, sia gli approcci content-based che quelli collaborativi utilizzano la stessa misura del coseno, ma, mentre nei sistemi di raccomandazione content-based essa è utilizzata per misurare la similarità tra vettori nei pesi TF-IDF, nei sistemi collaborativi misura la similarità fra vettori dell'effettivo voto specificato dall'utente.

Sono state proposte molte modifiche per migliorare le performance delle tecniche standard correlation-based e cosine-based, come *default voting*, *inverse user frequency*, *case amplification* e *weighted-majority prediction*.

Per esempio, il default voting è un'estensione degli approcci memory-based già descritti. È stato osservato che, ogni volta che ci sono relativamente pochi voti specificati dall'utente, questi metodi potrebbero non lavorare bene nel calcolo della similarità fra l'utente *x* ed *y*, dato che la misura di similarità è basata sull'intersezione degli itemset, vale a dire gli insiemi di item votati *sia* da *x* che da *y*. È stato empiricamente dimostrato che l'accuratezza della previsione dei voti potrebbe migliorare se si assumessero dei valori di default per i voti mancanti.

Inoltre, mentre le precedenti tecniche tradizionalmente sono state utilizzate per calcolare la similarità tra *utenti*, Sarwar et al. [17] hanno proposto di utilizzare le stesse tecniche correlation-based e cosine-based per calcolare la similarità tra *item* e dunque ottenendo i voti per essi. Questa idea è stata poi estesa per le migliori *N* raccomandazioni di item.

In aggiunta, gli studi di Sarwar hanno presentato la prova empirica che gli algoritmi basati su item possono fornire migliori performance computazionali rispetto ai metodi collaborativi user-based mentre, allo stesso tempo, forniscono qualità migliore o confrontabile con i migliori algoritmi basati su utente.

In contrasto coi metodi memory-based, gli algoritmi model-based utilizzano le collezioni di voti per apprendere un *modello,* che è poi utilizzato per costruire previsioni sui voti. Per esempio, Breese et al. [16] propongono un approccio probabilistico al collaborative filtering, dove i voti sconosciuti sono calcolati come segue:

$$r_{cs} = E(r_{cs}) = \sum_{i=0}^{n} i \times \Pr(r_{cs} = i \mid r_{cs'}, s' \in S_c) \qquad (14)$$

Si assume che i valori dei voti sono interi fra 0 ed *n* e l'espressione di probabilità è la probabilità che l'utente *c* fornirà un voto particolare all'item *s* dati i voti degli item precedentemente votati dall'utente.

Per stimare questa probabilità, Breese et al. Propongono due modelli probabilistici alternativi: modelli a cluster e reti Bayesiane.

Nel primo modello, gli utenti "like-minded" sono clusterizzati in classi. Data l'appartenenza alla classe dell'utente, i voti dello stesso si assumono essere indipendenti, cioè la struttura del modello è come quella del modello naive Bayesiano. Il numero di classi ed i parametri del modello sono appresi dai dati.

Il secondo modello rappresenta ogni item nel dominio come un nodo in una rete Bayesiana, dove gli stati di ogni nodo corrispondono ai possibili valori dei voti per ogni item. Sia la struttura della rete che le probabilità condizionate sono apprese dai dati. Una limitazione a questo approccio è che ogni utente può essere inserito in un singolo cluster, considerando che alcune applicazioni di raccomandazione possono beneficiare della possibilità di clusterizzare utenti in diverse categorie alla volta. Per esempio, in un'applicazione per la raccomandazione di libri, un utente potrebbe essere interessato di un argomento (ad esempio, programmazione) per scopi di lavoro e di un argomento completamente differente (ad esempio, pesca) per svago.

Inoltre, Billsus e Pazzani [18] hanno proposto un metodo di collaborative filtering in un framework di machine learning, dove possono essere

utilizzate diverse tecniche di machine learning (come reti neurali artificiali) accoppiate con tecniche di feature extraction (come la decomposizione a valori singolari, SVD – una tecnica algebrica per ridurre la dimensionalità delle matrici).

Sia Breese et al. che Billsus e Pazzani confrontano i loro rispettivi approcci model-based con approcci standard memory-based e constatano che, in alcune applicazioni, i metodi model-based superano quelli memory-based in termini di accuratezza delle raccomandazioni. Comunque, il confronto in entrambi i casi è puramente empirico e non fornisce alcuna prova teoretica a supporto dello stesso.

Ci sono stati diversi altri approcci alla raccomandazione collaborativa model-based proposti in letteratura. È stato proposto un modello statistico per il collaborative filtering [19] e sono stati confrontati molti algoritmi per stimare i parametri del modello, inclusi il *K-means clustering* e il campionamento di *Gibbs*. Altri metodi di collaborative filtering includono modello Bayesiano, modello probabilistico relazionale, regressione lineare, e il modello di massima entropia.

Più recentemente, è stato dato un significativo contributo di ricerca per cercare di modellare il processo di raccomandazione utilizzando modelli probabilistici più complessi. Per esempio, sono stati utilizzati processi decisionali di Markov per generare raccomandazioni, così come la *latent semantic analysis* e combinazioni di tecniche classiche.

Come nel caso di tecniche content-based, la principale differenza tra tecniche collaborative model-based ed approcci heuristic-based è che le tecniche model-based calcolano le previsioni di utilità (rating) basandosi non su regole euristiche ad hoc, ma, piuttosto, basandosi su un *modello*

appreso dai dati sottostanti utilizzando tecniche statistiche e di machine learning.

Esistono metodi che combinano gli approcci memory-based e model-based, ed è stato empiricamente dimostrato che l'uso di questi approcci combinati può fornire migliori raccomandazioni piuttosto che i puri approcci collaborativi memory e model based.

Un approccio differente per migliorare la performance degli algoritmi esistenti di collaborative filtering consiste nello scegliere accuratamente l'insieme di voti specificati dall'utente da utilizzare come input tramite tecniche specifiche, atte ad escludere rumore, ridondanza, e limitando la *sparsity* (scarsa densità) dei dati relativi ai voti. I risultati empirici dimostrano il miglioramento in accuratezza ed efficienza per gli algoritmi di collaborative filtering model-based.

Inoltre, fra tutti gli ultimi interessanti sviluppi, Yu et al. [20] propongono un approccio probabilistico al collaborative filtering che costituisce ancora un altro modo per combinare tecniche memory-based e content-based. In particolare, vengono fatte due proposte:

1. Utilizzare un approccio attivo all'apprendimento per migliorare il modello probabilistico di ogni preferenza dell'utente;

2. Utilizzare i profili utente memorizzati in un modello misto per calcolare le raccomandazioni.

Il secondo aspetto dell'approccio proposto sviluppa alcune delle idee utilizzate nei tradizionali algoritmi memory-based.

1.5 Limitazioni dei metodi Collaborativi

I sistemi di raccomandazione puramente collaborativi non hanno alcune delle carenze che i sistemi content-based hanno. In particolare, dato che i sistemi collaborativi usano le raccomandazioni di altri utenti (voti), essi permettono di far fronte a qualsiasi tipo di contenuto e di raccomandare qualsiasi item, persino quelli che sono dissimili a quelli visti in passato. Ad ogni modo, i sistemi collaborativi hanno le proprie limitazioni che verranno descritte nei prossimi sottoparagrafi.

1.5.1 Problema del Nuovo Utente

Questo è lo stesso problema dei sistemi content-based. Per restituire raccomandazioni accurate, il sistema deve dapprima apprendere le preferenze dell'utente dai voti che l'utente fornisce.

Diverse proposte sono state fatte per risolvere questo problema. La maggior parte di esse utilizzano l'approccio *ibrido* alla raccomandazione, che combina tecniche content-based e collaborative. Un approccio alternativo invece consiste nell'esplorazione di svariate tecniche per determinare i migliori (ad esempio, quelli con maggiore contenuto informativo nel sistema di raccomandazione) item per un nuovo utente.

Queste tecniche usano strategie che sono basate sulla popolarità dell'item, sull'entropia, sulla personalizzazione dell'utente, e combinazioni delle precedenti.

1.5.2 Problema del Nuovo Item

Nuovi item sono aggiunti con regolarità ai sistemi di raccomandazione. I sistemi collaborativi si basano unicamente sulle preferenze degli utenti per

fornire raccomandazioni. Perciò, finché il nuovo item non viene votato da un sostanziale numero di utenti, il sistema di raccomandazione non sarà in grado di raccomandarlo.

Questo problema può anche essere risolto utilizzando approccio ibridi di raccomandazione, descritti nei prossimi paragrafi.

1.5.3 Sparsity (Scarsa densità)

In ogni sistema di raccomandazione, il numero di voti già ottenuti è solitamente molto piccolo se confrontato al numero di voti che devono essere predetti. La previsione efficace di voti da un piccolo numero di esempi è molto importante; inoltre, il successo dei sistemi di raccomandazione collaborativi dipende dalla disponibilità di una massa critica di utenti.

Per esempio, in un sistema di raccomandazione di film, potrebbero esserci molti film votati da poche persone e questi film saranno raccomandati molto raramente, anche se questi pochi utenti abbiano dato ad essi voti alti. Inoltre, per gli utenti i cui gusti sono inusuali confrontati col resto della popolazione, non ci sarà alcun altro utente particolarmente simile, il che porta a raccomandazioni poco affidabili.

Un modo di superare il problema della sparsity di voti è di utilizzare le informazioni sul profilo utente quando si va a calcolare la similarità. Questo significa che due utenti potrebbero essere considerati simili non soltanto se essi hanno votati gli stessi film in maniera simile, ma anche se appartengono allo stesso segmento demografico. Per esempio, si possono utilizzare sesso, età, CAP, grado di istruzione, e informazioni sul lavoro degli utenti per la raccomandazione.

Un altro approccio che esplora anche la similarità tra utenti potrebbe essere quello di approcciare il problema della sparsity utilizzando algoritmi che esplorano le associazioni tra consumatori attraverso le loro precedenti transazioni e feedback.

Infine, un approccio più "matematico" potrebbe essere quello di ridurre la dimensione della matrice sparsa tramite decomposizione a valori singolari (SVD). La SVD è un metodo molto conosciuto per la fattorizzazione di matrici che fornisce la migliore approssimazione *low-rank* della matrice originale.

1.6 Metodi Ibridi

Diversi sistemi di raccomandazione utilizzano un approccio *ibrido* combinando metodi collaborativi e content-based, e questo aiuta ad evitare alcune limitazioni dei sistemi puri. I differenti modi di combinare i metodi collaborativi e content-based in un sistema di raccomandazione ibrido possono essere classificati come segue:

1. Implementare metodi collaborativi e content-based separatamente e combinare le loro previsioni;
2. Incorporare alcune caratteristiche content-based all'interno di un approccio collaborativo;
3. Incorporare alcune caratteristiche collaborative all'interno di un approccio content-based;
4. Costruire un modello generale di unificazione che incorpora caratteristiche sia content-based che collaborative.

Ciascuno dei precedenti approcci è stato utilizzato da molti ricercatori nel campo dei sistemi di raccomandazione; questo verrà descritto nei successivi sottoparagrafi.

1.6.1 Combinare Recommender separati

Un modo di costruire sistemi di raccomandazione ibridi consiste nell'implementare sistemi collaborativi e content-based separati. Dopo aver fatto ciò, si possono avere due differenti scenari.

In primo luogo, si possono combinare gli output (rating) ottenuti dai sistemi di raccomandazione individuali in una raccomandazione finale utilizzando o una combinazione lineare dei voti o uno schema di voti. In alternativa, si può utilizzare uno dei recommender individuali in ogni momento scegliendo di usare quello che è "migliore" degli altri basandosi su alcune metriche per la "qualità" della raccomandazione.

Per esempio, il sistema DailyLearner [12] seleziona il sistema di raccomandazione che può fornire le raccomandazioni con il più alto livello di confidenza, mentre altri sistemi potrebbero utilizzare il sistema le cui raccomandazioni siano maggiormente consistenti coi voti passati dell'utente.

1.6.2 Aggiungere caratteristiche Content-Based ai modelli Collaborativi

Diversi sistemi di raccomandazione ibridi, come ad esempio Fab [01], sono basati sulle tradizionali tecniche collaborative ma mantengono anche i profili content-based per ogni utente. Questi profili content-based, e non

invece gli item votati più comunemente, sono poi utilizzati per calcolare la similarità fra due utenti.

Questo metodo porta a superare alcuni problemi collegati alla sparsity di un approccio puramente collaborativo, poiché, tipicamente, non molte paia di utenti avranno un numero significativo di item votati in comune.

Un altro beneficio di questo approccio è che agli utenti potrà essere raccomandato un item non soltanto quando esso è votato con voti alti dagli utenti con profili simili, ma anche direttamente, cioè quando questo item ha un voto alto se relazionato al profilo utente.

Altri sistemi impiegano un approccio in qualche modo simile utilizzando una varietà di *filterbots* – agenti specializzati nella content analysis che agiscono come partecipanti aggiuntivi in una comunità di collaborative filtering.

Come risultato, gli utenti i cui voti concordano con alcuni dei voti dei filterbot saranno in grado di ricevere migliori raccomandazioni. In ultimo, alcuni sistemi utilizzano un approccio collaborativo in cui i tradizionali vettori dei voti dell'utente sono aumentati con voti addizionali, che sono calcolati utilizzando un predictor content-based.

1.6.3 Aggiungere caratteristiche Collaborative ai modelli Content-Based

Il più popolare approccio in questa categoria è di utilizzare alcune tecniche di *dimensionality reduction* su un gruppo di profili content-based. Per esempio, si può utilizzare il latent semantic indexing (LSI) per creare una visione collaborativa di una collezione di profili utente, dove questi sono rappresentati da vettori di termini (come discusso in *1.2*). Il

risultato è un miglioramento della performance, se confrontata con l'approccio puramente content-based.

1.6.4 Sviluppare un singolo modello di Raccomandazione Unificato

Molti ricercatori hanno seguito questo approccio negli anni recenti. Un esempio è quello di utilizzare caratteristiche content-based e collaborative (per esempio, l'età o il sesso degli utenti oppure il genere di film) in un unico classificatore rule-based. Altri approcci di questo genere sono:

- Utilizzare un metodo probabilistico unificato per combinare raccomandazioni content-based e collaborative, basandosi sul LSI probabilistico;
- Utilizzare modelli di regressione Bayesiani che impiegano il metodo Markov Chains Monte Carlo (MCMC) per la previsione e la stima dei parametri.

In particolare, Ansari et al. [21] utilizzano le informazioni del profilo di utenti ed item in un singolo modello statistico che stima i voti sconosciuti r_{ij} per l'utente *i* e l'item *j*:

$$\begin{aligned} r_{ij} &= x_{ij}\mu + z_i\gamma_j + w_j\lambda_i + e_{ij} \\ e_{ij} &\sim N(0,\sigma^2) \\ \lambda_i &\sim N(0,\Lambda) \\ \gamma_j &\sim N(0,\Gamma) \end{aligned} \qquad (15)$$

Dove $i = 1,...,I$ e $j = 1,...,J$ rappresentano, rispettivamente, utenti ed item, mentre e_{ij}, λ_i ed γ_j sono variabili random che tengono conto

rispettivamente di rumore, sorgenti non osservate ed eterogeneità dell'utente.

Inoltre, x_{ij} è una matrice che contiene caratteristiche di utenti ed item, z_i è un vettore di caratteristiche dell'utente e w_j è un vettore di caratteristiche dell'item. I parametri sconosciuti di questo modello sono $\mu, \sigma^2, \Lambda, \Gamma$ ed essi sono stimati dai dati relativi ai voti già noti utilizzando il metodo Markov Chains Monte Carlo (MCMC).

Riassumendo, questo approccio utilizza attributi dell'utente $\{z_i\}$ che costituiscono una parte del profilo utente, attributi dell'item $\{w_j\}$ che costituiscono un'altra parte del profilo utente, e le loro interazioni $\{x_{ij}\}$ per stimare il voto di un item.

I sistemi di raccomandazione ibridi possono anche essere migliorati tramite tecniche basate su conoscenza, come il ragionamento case-based, per portare un miglioramento all'accuratezza delle raccomandazioni e per eliminare alcune delle limitazioni (problemi del nuovo utente e del nuovo item).

1.7 Considerazioni finali sui Sistemi di Raccomandazione

Come descritto ai paragrafi *1.2*, *1.4* e *1.6*, molta ricerca è stata fatta sulle tecnologie di raccomandazione negli ultimi anni, e questa ha utilizzato una vasta gamma di tecniche statistiche, di machine learning, di information retrieval ed altre che hanno fatto significativamente avanzare lo stato dell'arte se confrontate coi primi sistemi di raccomandazione che utilizzavano euristiche collaborative e content-based.

Come è stato discusso in precedenza, i sistemi di raccomandazione possono essere suddivisi come facenti parte delle categorie:

1. Content-based
2. Collaborativi
3. Ibridi

Oppure

1. Heuristic-based
2. Model-based

(se ci si basa invece sulla tipologia di tecniche di raccomandazione utilizzate per stimare i voti)

È possibile, in questo modo, utilizzare due dimensioni ortogonali per classificare le ricerche sui sistemi di raccomandazione tramite una matrice 2x3 come in Tabella 2:

Approccio alla Raccomandazione	Tecniche di Raccomandazione	
	Heuristic-Based	**Model-Based**
Content-Based	Tecniche comunemente usate: • TF-IDF (information retrieval) • Clustering Esempi di ricerca rappresentativi: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Tecniche comunemente usate: • Classificatori Bayesiani • Clustering • Alberi decisionali • Reti neurali artificiali Esempi di ricerca rappresentativi: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Tecniche comunemente usate: • Nearest neighbor (coseno, correlazione) • Clustering • Graph theory Esempi di ricerca rappresentativi: • Resnick et al. 1994 • Hill et al. 1995 • Shardnand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitk 1999 • Sarwar et al. 2001	Tecniche comunemente usate: • Reti Bayesiane • Clustering • Reti neurali artificiali • Regressione lineare • Modelli probabilistici Esempi di ricerca rappresentativi: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Ibride	Combinare componenti content-based e collaborative utilizzando: • Combinazione lineare di voti predetti • Diversi schemi di voto • Incorporamento di una componente come parte dell'euristica per l'altro Esempi di ricerca rappresentativi: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combinare componenti content-based e collaborative tramite: • Incorporamento di una componente come parte del modello per l'altro • Costruzione di un modello unificato Esempi di ricerca rappresentativi: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Tabella 2 - Classificazione delle ricerche sui Sistemi di Raccomandazione

2. ITem Recommender e la sua reingegnerizzazione

In questo capitolo analizzeremo in dettaglio il processo di reingegnerizzazione di ITem Recommender, partendo da nozioni preliminari, passando per il vecchio sistema, per poi arrivare alle modifiche implementate per migliorare la qualità del sistema software.

2.1 Nozioni preliminari

Come già sottolineato, essere a conoscenza degli interessi di un utente può risultare un enorme vantaggio. Ad esempio, sarebbe possibile:

- filtrare i documenti eliminando quelli che non sono rilevanti;
- aiutare l'utente nella formulazione delle query;
- fornire un risultato che sia ordinato in modo da rispettare una stima del grado di rilevanza che le informazioni trovate hanno in funzione degli interessi dell'utente.

L'obiettivo di questo studio è costruire degli user model che raccolgano in modo più "fedele" possibile gli interessi dell'utente, il che si tradurrebbe poi in un aumento dell'efficacia di un sistema di ricerca.

In tal modo si passerebbe da un sistema di IR classico ad uno definito dalla quintupla:

$$< D, Q, F, R(d_i, q_j), P >$$

dove:

- *D* è l'insieme di tutte le rappresentazioni dei documenti nella collection;
- *Q* è l'insieme contenente le rappresentazioni delle informazioni desiderate dall'utente, le query;
- *F* è il framework necessario per il trattamento dei documenti, delle query, delle relazioni tra loro e delle loro rappresentazioni.
- *R* è una funzione di *relevance* che associa ad una rappresentazione del documento d_i in *D* e una di query q_j in Q, un numero reale indice della conformità del documento alla query;
- *P* è il profilo utente, che rappresenta le preferenze di un utente durante il processo di retrieval.

Successivamente si presenterà anche il dataset EachMovie: sebbene il sistema sia stato modificato per lavorare su qualunque database, previa configurazione di un file di proprietà, il vincolo principale oggetto di modifica del sistema è quello di poter lavorare solo sugli slot presenti nei documenti memorizzabili in questa base di dati. Tutte le figure, gli esempi riportati in questa sezione (nella descrizione dei profili) ed in quelle successive si riferiscono a questo dataset per cui è sembrato necessario fornirne una breve descrizione.

2.2 Creazione di profili come problema di categorizzazione del testo

Apprendere gli interessi degli utenti, trovare informazioni rilevanti nel web, guidare la ricerca dell'utente, ordinare la posta elettronica, catalogare articoli di newsgroup e pagine Web, sono tutte operazioni che possono essere portate a termine con l'utilizzo di algoritmi di classificazione.

La teoria del TC (Text Categorization) nasce agli inizi degli anni '60 ma solo negli anni '90, è diventata uno dei maggiori campi di interesse nel settore della computer science. [22]

Lo scopo della categorizzazione del testo è la classificazione dei documenti in un numero prefissato e predefinito di categorie. In generale ogni documento può appartenere ad una categoria, a più categorie o a nessuna categoria: si cerca di definire dei classificatori che siano in grado di assegnare automaticamente una categoria ad un nuovo documento.

Formalmente, detti *D* l'insieme dei documenti e *C* l'insieme delle categorie predefinite, la categorizzazione del testo è una funzione

$$\Phi' : D \to C$$

che assegna un valore booleano ad ogni coppia $< d_j, c_i > \in D \times C$.

In particolare, si ottiene *true* se c_i viene assegnata a d_j, *false* in caso contrario. L'intento è approssimare, con maggiore precisione possibile, la funzione non nota

$$\Phi : D \to C$$

che descrive come i documenti debbano essere classificati. [22]

Il passo successivo è rappresentato dalla misura dell'efficacia della classificazione generata, ottenibili in termini di precisione, richiamo e accuratezza.

In ogni modo, in base alle caratteristiche intrinseche del problema da analizzare, potrebbe essere necessario dover assegnare ad ogni documento d_i:

1. una sola categoria c_j ed in questo caso si parlerà di single-label case, oppure
2. una o più categorie c_j $(0 < j < |C|)$ ed in questo caso si parlerà di multi-label case.

Nel sistema realizzato, il problema di costruire profili utente è stato considerato come un caso particolare di *single-label* TC, ovvero come un problema di categorizzazione binaria.

Formalmente, ogni documento $d_j \in D$ viene assegnato alla categoria c_j oppure al suo complementare $\bar{c}_j$. Più semplicemente, ogni documento viene classificato come interessante o non interessante in base alle preferenze dell'utente e questo riduce a due il numero di classi: *c*+ (user likes) e *c*- (user dislikes).

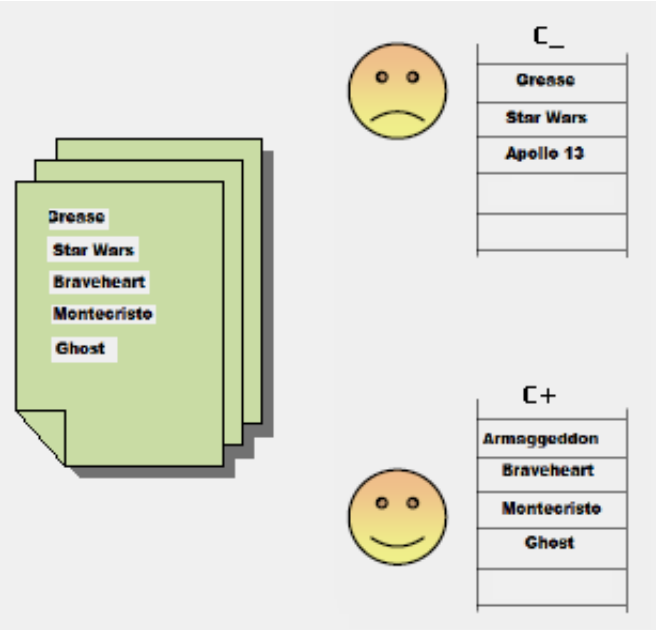

Figura 1 - Esempio di text categorization

In verità, si possono individuare due principali approcci al problema di categorizzazione del testo. Il primo pone le basi sui principi dell'*ingegneria della conoscenza*, per cui consta della definizione manuale di un set di regole inserite in un sistema esperto. Il secondo, invece, adopera tecniche di *machine learning* grazie alle quali, a partire da un set preclassificato di documenti (*training documents*), si riesce a costruire un classificatore capace di assegnare la classe appropriata ad un nuovo documento.

Molti sistemi sono stati sviluppati con il *secondo approccio*, questo perché il dominio dei documenti testuali coinvolge grandi quantità di dati che evolvono continuamente e non esiste una teoria che definisca come debba essere classificato un documento. L'approccio dell'apprendimento automatico risulta il più adatto a trattare questo compito e, in questo, la visione *bayesiana* della statistica fornisce una metodologia appropriata.

2.3 Il metodo Naive Bayes

Di recente sono stati studiati vari classificatori bayesiani con dei risultati piuttosto sorprendenti: nonostante la loro estrema semplicità, riescono a raggiungere delle prestazioni addirittura superiori rispetto ai più noti algoritmi di apprendimento. Queste metodologie di tipo probabilistico fanno delle forti assunzioni sulla generazione dei dati ed utilizzano un modello probabilistico che ingloba queste assunzioni.

Avvalendosi di un insieme di dati etichettati per l'addestramento, si stimano i parametri del modello generativo e si classificano le nuove istanze utilizzando il teorema di Bayes e selezionando la classe o categoria che ha la probabilità più alta di aver generato l'esempio. [23]

Il classificatore Naive Bayes è un metodo di apprendimento bayesiano che si è rivelato utile in molte applicazioni, tra cui la categorizzazione dei documenti testuali. Analizziamo il metodo più in dettaglio.

2.3.1 Nozioni di probabilità

Noti i principi base della teoria della probabilità, quali la probabilità di un evento, certo ed impossibile, e la probabilità di eventi congiunti:

$$0 \leq P(Evento) \leq 1$$
$$P(Evento) = 1 \quad \textit{se l'evento è certo}$$
$$P(Evento) = 0 \quad \textit{se l'evento è impossibile}$$
$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

È possibile definire il concetto di probabilità condizionata:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Che indica la probabilità che si verifichi un evento A sapendo che si è verificato B.

In particolare, se i due eventi A e B sono indipendenti allora:

$$P(A \cap B) = P(A) \cdot P(B)$$

Per cui:

$$P(A \mid B) = P(A) \quad e \quad P(B \mid A) = P(B)$$

In questo contesto, il *teorema di Bayes* afferma che:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

2.3.2 Classificazione con metodo Bayesiano

In virtù delle relazioni appena definite, detto $C = \{c_1, c_2, ..., c_n\}$ l'insieme delle categorie in cui i documenti si possono trovare e *d* la descrizione di un'istanza da classificare, per determinare a quale categoria appartiene *d* è necessario calcolare

$$\forall c \in C \quad P(c_i \mid d) = \frac{P(c_i) \cdot P(d \mid c_i)}{P(d)} \qquad (1)$$

ed il modello assegnerà il documento *d* alla classe che rende massima questa relazione.

Si osserva che $P(d)$ è facilmente calcolabile sfruttando la probabilità a priori delle varie c_i:

$$P(d) = \sum_{i=1}^{n} P(c_i) \cdot P(d \mid c_i)$$

ma in genere, il termine viene trascurato, considerando che risulta uguale per tutte le categorie c_i. Invece, le probabilità a priori delle singole categorie, $P(c_i)$, e la probabilità condizionata del documento data la categoria, $P(d \mid c_i)$, vengono stimate osservando i dati di training.

È facile osservare che i dati di training possono non essere sufficienti a definire una stima corretta. L'inconveniente viene superato con l'assunzione che, data una classe, tutte le parole, i *token* del documento *d*, sono tra loro tra loro "*condizionalmente*" *indipendenti*. [22] Questa affermazione viene detta *assunzione "Naive Bayes"* e, sebbene sia violata nella maggior parte dei problemi reali, il metodo si comporta molto bene e risulta essere molto efficace. [23]

Il calcolo della probabilità $P(d \mid c_i)$ viene effettuato secondo il *multinomial event model*. Questo modello tratta il documento come un vettore *V* nel

quale si tiene traccia di quante volte una parola appare nel documento. Così la probabilità condizionata $P(d \mid c_i)$ si ottiene calcolando:

$$P(d \mid c_i) = \prod_{w \in V_{di}} P(t_k \mid c_i)^{N(d_i, t_k)} \qquad (2)$$

In cui:

- $N(d_i, t_k)$ rappresenta le occorrenze del termine t_k nel documento d_i
- V_{di} rappresenta il vettore che contiene le parole che appaiono nel documento d_i

Quindi, in base alla (1), la probabilità a posteriori del documento *D* appartenente alla classe ci si calcola come:

$$P(c_i \mid d) = P(c_i) \cdot \prod_{w \in V_{di}} P(t_k \mid c_i)^{N(d_i, t_k)}$$

Esempio:

$C = \{piace, \neg piace\}$
$D = programmazione \wedge java2SE \wedge web$

Date le probabilità a priori:

	Piace	**Non Piace**
$P(c_i)$	0,7	0,3
$P(programmazione \mid c_i)$	0,3	0,7
$P(java2SE \mid c_i)$	0,2	0,8
$P(web \mid c_i)$	0,7	0,3

Tabella 3 - Esempio di probabilità a priori

Si possono calcolare:

$$P(piace \mid d) = \frac{P(piace) \cdot P(d \mid piace)}{P(d)} =$$

$$= \frac{0,7 \cdot (0,3 \cdot 0,2 \cdot 0,7)}{0,7 \cdot (0,3 \cdot 0,2 \cdot 0,7) + 0,3 \cdot (0,7 \cdot 0,8 \cdot 0,3)} = 0,368$$

$$P(\neg piace \mid d) = \frac{P(\neg piace) \cdot P(d \mid \neg piace)}{P(d)} =$$

$$\frac{0,3 \cdot (0,7 \cdot 0,8 \cdot 0,3)}{0,7 \cdot (0,3 \cdot 0,2 \cdot 0,7) + 0,3 \cdot (0,7 \cdot 0,8 \cdot 0,3)} = 0,632$$

In questo esempio il modello "prevede" che l'istanza *non piaccia* all'utente. Questo sembra ragionevole, considerando che all'utente piace il tema *web* in generale, ma non apprezza ciò che riguarda l'informatica

tecnica. Inoltre, osservando che il documento in questione *d* presentava proprio queste features (java, programmazione e web), la previsione sembra corretta.

2.3.3 Stima delle probabilità delle parole

Il passo successivo prevede il calcolo del termine

$$P(t_k \mid c_i)$$

che rappresenta la stima della probabilità del termine t_k data la categoria c_i.

Purtroppo, il calcolo delle probabilità a priori è soggetto ad alcuni errori: se ad esempio si considera una feature rara d_k, che nelle istanze del training set non si presenta mai, si avrà:

$$\forall c_i : P(d_k \mid c_i) = 0$$

e quando poi in fase di test si dovesse presentare un documento *d* che ha la caratteristica d_k si otterrebbe:

$$\forall c_i : P(d \mid c_i) = 0 \quad \text{e} \quad \forall c_i : P(c_i \mid d) = 0 .$$

Per porre rimedio a questo problema si potrebbe utilizzare la *stima di Laplace* oppure la *formula Written-Bell*. Più precisamente, Written-Bell determina $P(t_k \mid c_i)$ secondo le seguenti relazioni:

$$P(t_k \mid c_i) = \begin{cases} \dfrac{N(t_k, c_j)}{V_{cj} + \sum_i N(t_i, c_j)} & se \quad N(t_k, c_j) \neq 0 \\ \dfrac{V}{V_{cj} + \sum_i (t_i, c_j)} \cdot \dfrac{1}{V - V_{cj}} & se \quad N(t_k, c_j) = 0 \end{cases}$$

In cui:

- $N(t_k, c_j)$ è l'occorrenza del termine t_k nei dati di addestramento per la classe c_j;
- V_{cj} è il numero di parole distinte nella classe c_j;
- V è il numero di parole distinte di tutte le classi.

2.4 ITem Recommender

Il progetto ITem Recommender è un sistema basato sulla profilazione utente, che mira a migliorare la relazione fornitore/acquirente nell'ambito del commercio elettronico, attraverso la realizzazione di agenti intelligenti.

Questo sistema, realizzato presso il laboratorio *LACAM* del Dipartimento di Informatica presso l'Università degli studi di Bari, ingloba tecniche di *Information Filtering Content Based* e di *Collaborative Filtering*.

ITR è un'opera che ha l'intento di fornire mezzi tecnici e organizzativi per migliorare le relazioni cliente-fornitore nell'e-Commerce del futuro, per

cui lo scopo principale è quello di incrementare il comfort nell'uso di questi siti, nonché fornire una certa flessibilità nella modalità di interfacciarsi con i clienti.

Il sistema si basa su agenti intelligenti che rappresentano assistenti virtuali in grado di fornire un supporto personalizzato agli utenti. Tali agenti possono istruire i clienti nell'utilizzo di un sito Web, evidenziare nuove offerte, aiutarli a vagliare i prodotti disponibili e fornire assistenza di vario tipo.

Il dominio applicativo iniziale riguardava libri, CD e DVD; in ogni caso, la tecnologia basata su agenti intelligenti può essere applicata in un contesto più ampio nell'ambito del commercio elettronico.

In questo progetto si possono individuare tre componenti fondamentali:

- *Recommender*: il suo compito è quello di interfacciare il profilo utente con l'IR-engine durante la fase di *personalizzazione* dei risultati. Tiene traccia dell'intera configurazione del sistema, utenti, documenti, voti, categorie e profili , tutte informazioni necessarie al momento di effettuare la vera e propria predizione di gradimento su un certo item.

- *BayesClassifier*: implementa il *classificatore di Bayes*. Il suo compito fondamentale è la creazione/aggiornamento dei profili con il metodo descritto al paragrafo precedente: questo processo avviene con l'interazione utente-sistema ed è fondamentale per il miglioramento delle prestazioni del sistema. L'altra funzione è quella di eseguire un processo di *Text Categorization* quando è richiesta la riclassificazione di un certo item, per conoscere se appartiene alla categoria "piace" o a quella "non piace".

- *ProfileExtractor*: il suo compito è memorizzare e rappresentare le informazioni derivanti dal dialogo con l'utente che costituiscono i profili sui quali basare le interazioni dell'agente con l'utente al quale il profilo si riferisce. Ha perciò la funzione di interfacciare il repository dei profili con il resto del sistema.

Nel sistema ITR è implementato uno user-model con il metodo Bayesiano presentato nel paragrafo precedente. Questo sfrutta le probabilità a priori di ciascuna categoria ottenute attraverso una adeguata fase di addestramento e su questa base, per ciascun documento da classificare, il sistema è in grado di ottenere una probabilità a posteriori sulle categorie.

Tutte le istanze sono classificate sulla base di voti espressi dall'utente, ovvero detto *r* il rate per una particolare istanza:

$$r \in c_{-} \quad se \quad \min \leq r < \max/2$$
$$r \in c_{+} \quad se \quad \max/2 \leq r \leq \max$$

Inoltre, ogni voto r dell'utente viene normalizzato per ottenere valori compresi tra 0 ed 1 nel seguente modo:

$$\omega_{+}^{i} = \frac{r_i - 1}{MAX - 1}$$
$$\omega_{-}^{i} = 1 - \omega_{+}^{i}$$

dove *MAX* indica il valore più alto del range.

Quindi, dato un utente *u* ed un *set di items* votati in una specifica categoria di interesse, il sistema definisce un profilo capace di riconoscere gli items rilevanti per u in quella categoria.

In base a quanto detto nel paragrafo 2.3.2, la probabilità condizionata di una categoria c_j in base ad una determinata istanza d_i viene definita in base alla formula:

$$P(c_j \mid d_i) = \frac{P(c_j)}{P(d_i)} \cdot \prod_{m=1}^{|S|} \prod_{k=1}^{|b_{im}|} P(t_k \mid c_j, s_m)^{n_{kim}}$$

In cui:

- $S = \{s_1, s_2, ..., s_n\}$ è l'insieme di slot;
- b_{im} è la Bag Of Words dell'istanza d_i nello slot s_m;
- n_{kim} è il numero di occorrenze del token t_k in b_{im}.

Analogamente a quanto detto in 2.3.2, il termine $P(d_i)$ è una costante, per cui il tutto si riduce a calcolare $P(c_j)$ e $P(t_k \mid c_j, s_m)$. Le probabilità a priori delle classi sono calcolate secondo:

$$\hat{P}(c_j) = \frac{\sum_{i=1}^{|TR|} \omega_j^i + 1}{|TR| + 2}$$

dove *TR* è il training set. Infine, in seguito alla considerazione che i documenti sono suddivisi in slot, le stime di Written-Bell sono state adeguate come segue:

$$\widehat{P}(t_k \mid c_j, s_m) = \begin{cases} \dfrac{N(t_k, c_j, s_m)}{V_{cj} + \sum_{i=1}^{|V_{cj}|} N(t_k, c_j, s_m)} & se \quad N(t_k, c_j, s_m) \neq 0 \\ \dfrac{V}{V_{cj} + \sum_{i=1}^{|V_{cj}|} N(t_k, c_j, s_m)} \cdot \dfrac{1}{V - V_{cj}} & se \quad N(t_k, c_j, s_m) = 0 \end{cases}$$

Dove:

- V_{ci} è il numero di parole distinte nella categoria c_i;
- V è il numero di parole distinte in tutte le classi;
- $N(t_k, c_j, s_m)$ è il conto delle occorrenze pesate della parola t_k per la classe c_j nello slot s_m ed è calcolata come

$$N(t_k, c_j, s_m) = \sum_{i=1}^{|TR|} \omega_j^i \cdot n_{kim}$$

2.5 Il dataset EACHMOVIE

Il progetto EachMovie è stato condotto dal CSRC (Compaq System Research Center) per 18 mesi nel biennio 1996/1997. In questo lasso di tempo, è stato collezionato un grande dataset di film votati dagli utenti, che comprende 2.811.983 voti per 1.628 film assegnati da 72.916 utenti. I film sono votati su una scala che varia da 0 a 5. Questi voti sono stati successivamente traslati nell'intevallo [0,1]. Il dataset originale non conteneva nessuna informazione riguardo il contenuto dei film, quindi è

stato necessario recuperare queste informazioni da Internet Movie Database. Le informazioni sono state recuperate tramite un semplice crawler utilizzando i link a IMDb forniti nel dataset originale.

Figura 2 – IMDb

In particolare, il crawler estrapolava informazioni riguardanti il titolo, il regista, il cast e la trama di ogni film. Inoltre veniva reperita anche una lista di *keyword* e la/le categoria/e (genere) cui ciascun film appartiene. I dati riguardanti il contenuto dei film sono stati successivamente inseriti in un file di testo CVS (comma separated value) e poi organizzati in un database relazionale. [22] Il database originale è stato poi sottoposto ad una fase di text-processing durante la quale sono state effettuate operazioni di *tokenizzazione* (suddivisione del testo in parole), *eliminazione delle stopwords* (congiunzioni, parole comuni, ecc.), *stemming* (ricondurre le parole alla loro radice), *POS-Tagging* (identificazione del "tipo" di parola – nome, avverbio, ecc.), *disambiguazione* (individuare la "semantica" di una frase), *count*, *tf_idf*.

Si noti infine che i film sono suddivisi nelle seguenti categorie: *azione, animazione, art_foreign, classici, commedie, drammatici, per la famiglia, di orrore, romantici e thriller*.

2.6 Reingegnerizzazione di ITR

La reingegnerizzazione di ITR è partita, in primo luogo, dalla scelta dei pattern di progetto da utilizzare all'interno del nuovo sistema. Dopo questo passo, è stato rimodellato il database per far fronte alle nuove esigenze del sistema.

Una volta definiti i pattern ed il database, obiettivo dei prossimi paragrafi sarà quello di mostrare la nuova applicazione reingegnerizzata, in termini di package, classi e metodi.

2.6.1 Trasferimento dei dati: Data Transfer Object

Per il trasferimento dei dati tra i diversi strati dell'applicazione si è decido di utilizzare il pattern Data Transfer Object.

Precedentemente conosciuto come Value Objects, il DTO è un pattern di progetto usato per trasferire dati tra sottosistemi di una stessa applicazione.

Per le operazioni di modifica dati, una soluzione plausibile sarebbe quella di passare un certo numero di parametri ad un metodo di invocazione remota, dati atti a descrivere la modifica stessa; questa soluzione risulta purtroppo in un insieme di inconvenienti, quali, ad esempio, un eccessivo carico di dati da passare via rete, quindi molteplici chiamate remote, e

un’inconsistenza strutturale soggetta alle possibili variazioni (future) dei parametri necessari.

In un’architettura distribuita la comunicazione avviene tra client ed EJB (session, entity, message-driven).

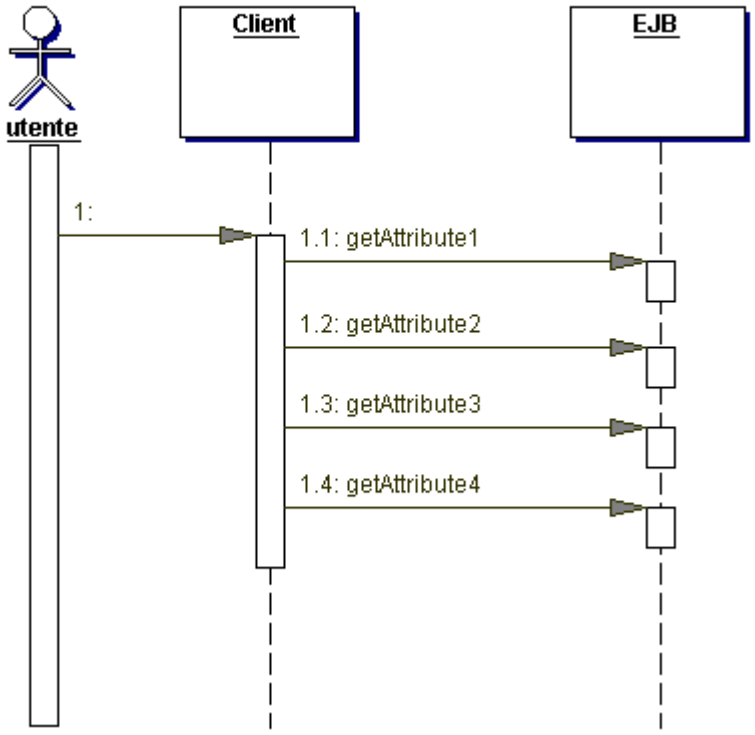

Figura 3 - Il client comunica direttamente con l’EJB passandogli i parametri

Nella figura precedente, il client effettua diverse chiamate di rete, richiedendo la serializzazione e deserializzazione dei valori di ritorno, bloccando il client nell’attesa che il server intercetti la chiamata, che vengano effettuati i check di sicurezza e delle transazioni, e che terminino i processi necessari al ritorno dei valori in questione. Inoltre, ad ogni

chiamata corrisponde, solitamente, una transazione differente; tutto questo risulta in uno scarso rendimento delle prestazioni.

Per evitare questi inconvenienti si ricorre alla scrittura di una semplice classe Java, chiamata Data Transfer Object, che incapsula i dati di interesse in un contenitore di facile trasporto in rete.

Ad esempio:

```
Import java.io.Serializable;

public class DTO implemets Serializable{

private String attribute1;
private String attribute2;
private String attribute3;
private long attribute4;

public String getAttribute1(){}
public String getAttribute2(){}
public String getAttribute3(){}
public long getAttribute4(){}

}
```

Il DTO (acronimo per Data Transfer Object) è utile, come già accennato, sia per operazioni di lettura che per operazioni di modifica dati in sistemi distribuiti.

Ogni cambiamento dei parametri utili a tali operazioni si riflette in un cambiamento degli attributi (e relativi metodi) nella classe DTO, quindi in un'unica classe.

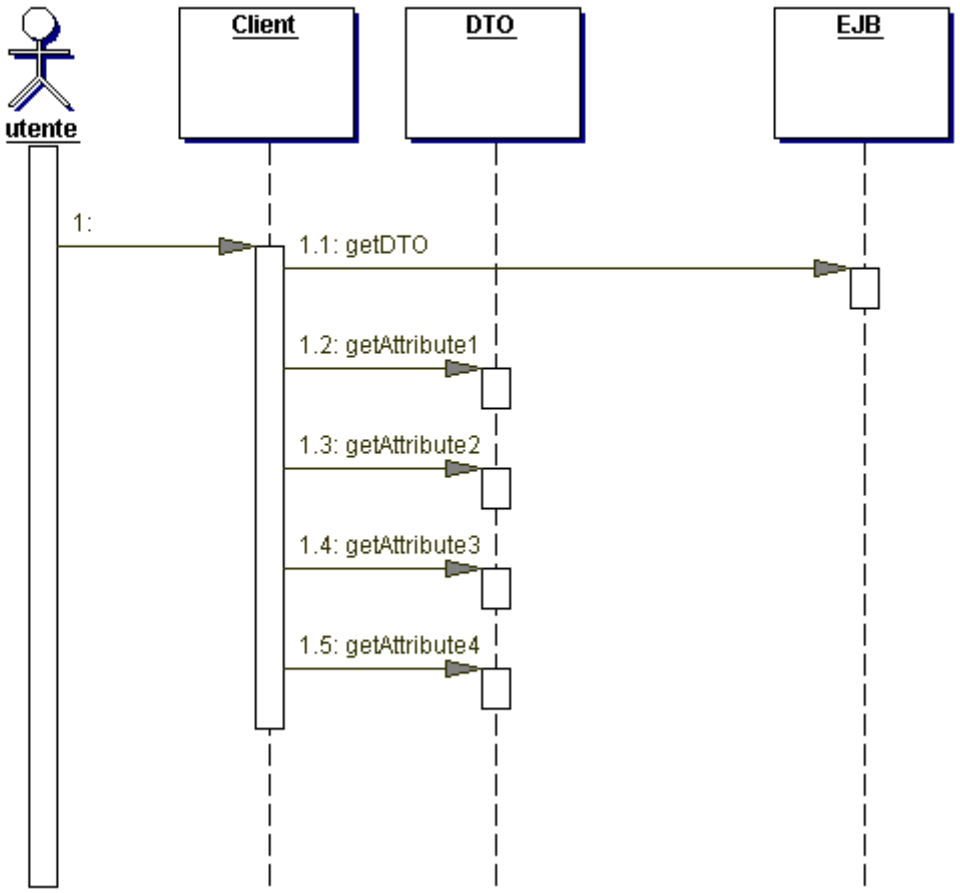

Figura 4 - Pattern DTO

La scelta della granularità del DTO (quanti attributi incapsularvi) e della sua necessità in un progetto non sono di facile analisi all'inizio del progetto stesso. In ITR, si è deciso di *mappare le tabelle del database singolarmente con un DTO apposito*.

Si tenga presente che in un progetto si susseguono continui cambiamenti (struttura oggetti, architettura, ecc.): ciò implica che le classi DTO sono anch'esse soggette a cambiamenti.

2.6.2 Gestione del database: Java DataBase Connectivity

JDBC (Java DataBase Connectivity), è un connettore per database che consente l'accesso alle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato. È costituita da una API, raggruppata nel package *java.sql*, che serve ai client per connettersi a un database. Fornisce metodi per interrogare e modificare i dati. È orientata ai database relazionali ed è Object Oriented. La piattaforma Java 2 Standard Edition contiene le API JDBC, insieme all'implementazione di un bridge JDBC-ODBC, che permette di connettersi a database relazionali che supportino ODBC. Questo driver è in codice nativo e non in Java.

L'architettura di JDBC, così come quella di ODBC, prevede l'utilizzo di un "*driver manager*", che espone alle applicazioni un insieme di interfacce standard e si occupa di caricare a "run-time" i driver opportuni per "pilotare" gli specifici DBMS. Le applicazioni Java utilizzano le "JDBC API" per parlare con il JDBC driver manager, mentre il driver manager usa le JDBC driver API per parlare con i singoli driver che pilotano i DBMS specifici. Esiste un driver particolare, il "JDBC-ODBC Bridge", che consente di interfacciarsi con qualsiasi driver ODBC in ambiente Windows.

Esistono driver free e commerciali per la maggior parte dei server di database relazionali. I driver possono essere di quattro tipi:

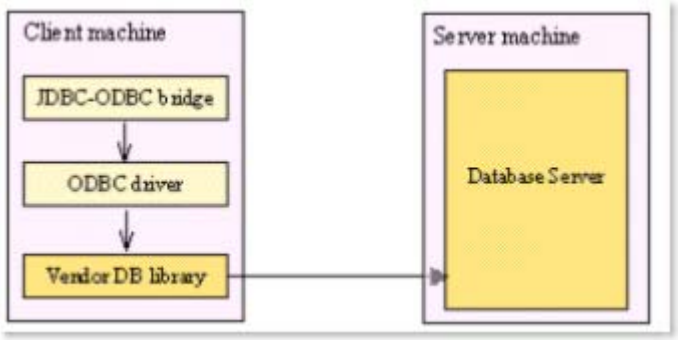

Figura 5 - Tipo 1: JDBC-ODBC bridge

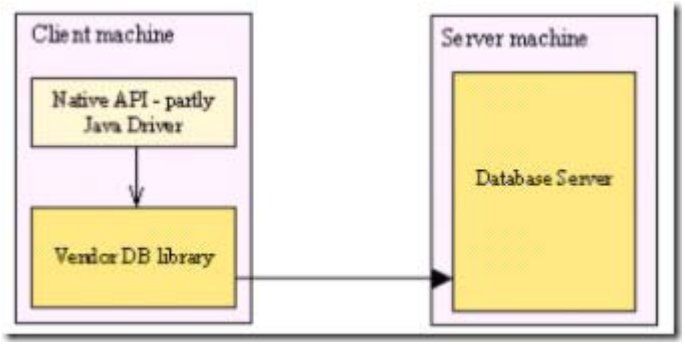

Figura 6 - Tipo 2: API nativa

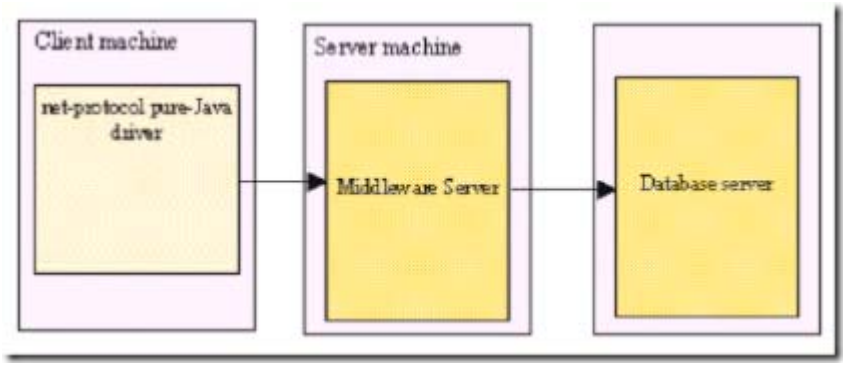

Figura 7 - Tipo 3: protocollo di rete

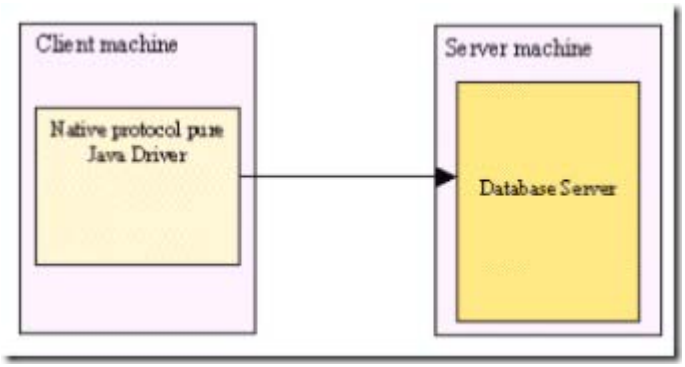

Figura 8 - Tipo 4: protocollo nativo

JDBC ammette che esistano diverse implementazioni e vengano utilizzate dalla stessa applicazione. L'API fornisce un meccanismo che carica dinamicamente i driver appropriati e li registra nel JDBC Driver Manager. Esso funge da fabbrica di connessioni.

Le connessioni JDBC supportano la creazione e l'esecuzione delle istruzioni. Esse possono essere comandi SQL come INSERT, UPDATE, DELETE, interrogazioni come SELECT o chiamate a stored procedure. I tipi di istruzioni supportati sono:

- *Statement:* l'istruzione viene inviata al database di volta in volta;
- *Prepared Statement:* l'istruzione viene compilata una sola volta, in modo che le chiamate successive siano più efficienti;
- *Callable Statement:* usati per chiamare le stored procedure.

I comandi di scrittura come INSERT, UPDATE e DELETE restituiscono un valore che indica quante righe sono state affette (inserite, modificate, cancellate) dall'istruzione. Essi non restituiscono altre informazioni.

Le interrogazioni (query) restituiscono un result set (classe *ResultSet*). È possibile spostarsi nel result set riga per riga (tramite il metodo *next()*). Si può accedere alle colonne di ogni singola riga chiamandole per nome o

per numero. Il result set può essere costituito da un numero qualsiasi di righe. Esso comprende dei metadati che indicano il nome, il tipo e le dimensioni delle colonne.

Esiste un'estensione di JDBC che permette, tra le altre cose, l'uso di result set scorribili e di cursori lato client.

Tutti i metodi delle API JDBC possono lanciare eccezioni, in quanto la connessione al DBMS in ogni momento può subire una interruzione o comunque si possono verificare errori nell'esecuzione dei comandi SQL. Tutte le eccezioni di JDBC derivano dalla classe SQLException e possono anche essere concatenate tra loro più eccezioni diverse. Ogni eccezione contiene un messaggio descrittivo, una stringa contenente lo stato SQL (conforme a quanto indicato nella specifica XOPEN SQL) e un intero contenente un codice errore addizionale specifico per il particolare driver o sorgente utilizzati.

Il rilascio delle risorse allocate durante le operazioni su database, in particolare l'oggetto *connection*, è particolarmente critica, in quanto il numero totale delle connessioni disponibili è limitato e normalmente la connessione al DB non viene rilasciata automaticamente quando non è più utilizzata. Per essere sicuri che una connessione sia chiusa correttamente, anche in caso di eccezione, conviene utilizzare la finally. Inoltre bisogna prestare particolare attenzione a non sollevare ulteriori eccezioni nel blocco finally.

Nel nuovo ITR, analogamente a quanto scelto per i DTO, si è deciso di *mappare le tabelle del database singolarmente con un JDBC apposito*.

2.6.3 Svincolarsi dalla staticità degli slots

Uno dei problemi principali di ITem Recommender era quello relativo alla staticità degli slots componenti l'item. Questo significa che, nella precedente visione del sistema, un item poteva essere composto soltanto dagli slot di default della tabella Item originale:

- Title;
- Cast;
- Director;
- Trama;
- Keyword.

È ovviamente intuibile che l'ambito applicativo del sistema originale era quello della raccomandazione di film.

La prima scelta progettuale di una certa importanza è stata quella di svincolarsi da un ambito particolare per permettere di impostare in maniera dinamica gli slot componenti l'item. Questo è stato fatto aggiungendo una nuova riga nel file di configurazione predefinito, *itr.properties*, elencando gli slot relativi all'item:

```
slots=title;cast;director;trama;keyword
```

E facendo in modo che, all'invocazione di particolari metodi di settaggio degli stessi slots, vengano invocate chiamate per la ricostruzione dinamica della tabella degli item stessi.

2.6.4 Modello del database

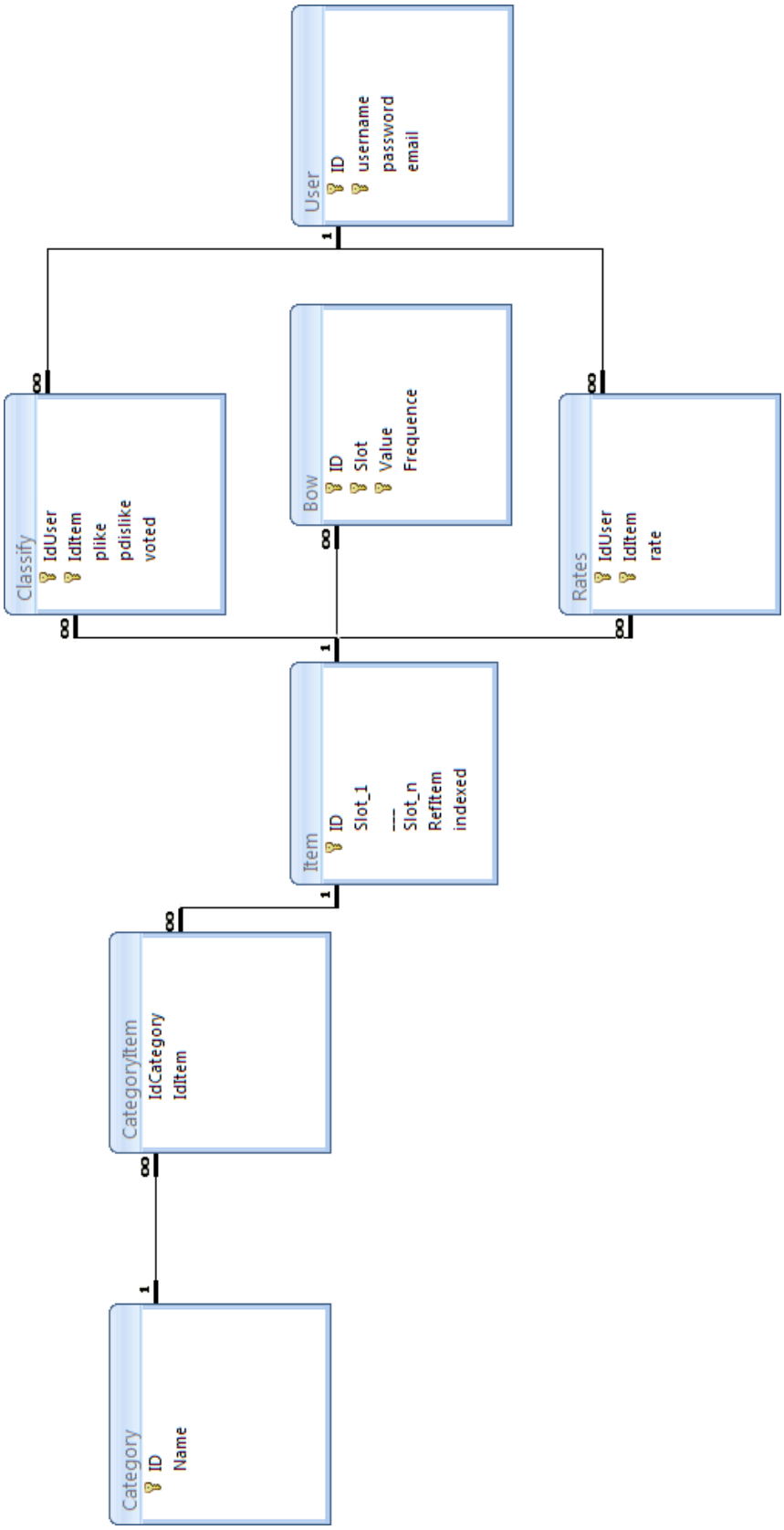

Figura 9 - Modello del database

In figura è riportato il nuovo modello del database, svincolato, come si è detto, dalla staticità relativa agli item. Vengono adesso riportate delle tabelle esplicative delle diverse componenti del database.

Bow

La tabella Bow mappa le Bag Of Words degli item, vale a dire le coppie parola-frequenza per ogni slot dell'Item

Nome	Tipo	Descrizione
ID	Integer	Identificativo dell'ID dell'item
Slot	Testo	Nome dello slot dell'item
Value	Testo	Parola presente nello slot
Frequence	Integer	Frequenza della parola nello slot

Tabella 4 - Bow

Category

La tabella Category raccoglie le categorie a cui gli Item possono appartenere

Nome	Tipo	Descrizione
ID	Integer	Identificativo della categoria
Name	Testo	Nome della categoria

Tabella 5 - Category

CategoryItem

La tabella CategoryItem mantiene la corrispondenza tra Item e Categoria/e a cui esso appartiene

Nome	Tipo	Descrizione
ID	Integer	Identificativo della categoria
Name	Testo	Nome della categoria

Tabella 6 - CategoryItem

Classify

La tabella Classify memorizza le classificazioni relative a ciascun utente

Nome	Tipo	Descrizione
IdUser	Integer	Identificativo dell'utente
IdItem	Integer	Identificativo dell'Item
plike	Double	Probabilità (score) che l'item sia gradito dall'utente
pdislike	Double	Probabilità (score) che l'item non sia gradito dall'utente
voted	Boolean	Flag che indica se l'item è stato già votato dall'utente

Tabella 7 - CategoryItem

Item

La tabella Item memorizza il contenuto degli item, suddivisi per slot

Nome	Tipo	Descrizione
ID	Integer	Identificativo dell'item
Slot_1	Testo	Contenuto dello slot *1*
…		
Slot_n	Testo	Contenuto dello slot *n*
RefItem	Testo	Riferimento esterno all'item (eventuale identificativo esterno)
Indexed	Boolean	Flag che indica se l'item è stato indicizzato (nella tabella Bow) o meno.

Tabella 8 - Item

Rates

La tabella Rates memorizza i voti già dati dagli utenti

Nome	Tipo	Descrizione
IdUser	Integer	Identificativo dell'utente
IdItem	Integer	Identificativo dell'item
Rate	Double	Voto dato dall'utente all'item

Tabella 9 - Rates

User

La tabella User memorizza gli utenti del sistema

Nome	Tipo	Descrizione
ID	Integer	Identificativo dell’utente
Username	Testo	Nome dell’utente
Password	Testo	Password dell’utente
Email	Testo	Email dell’utente

Tabella 10 - User

2.7 Diagramma dei package

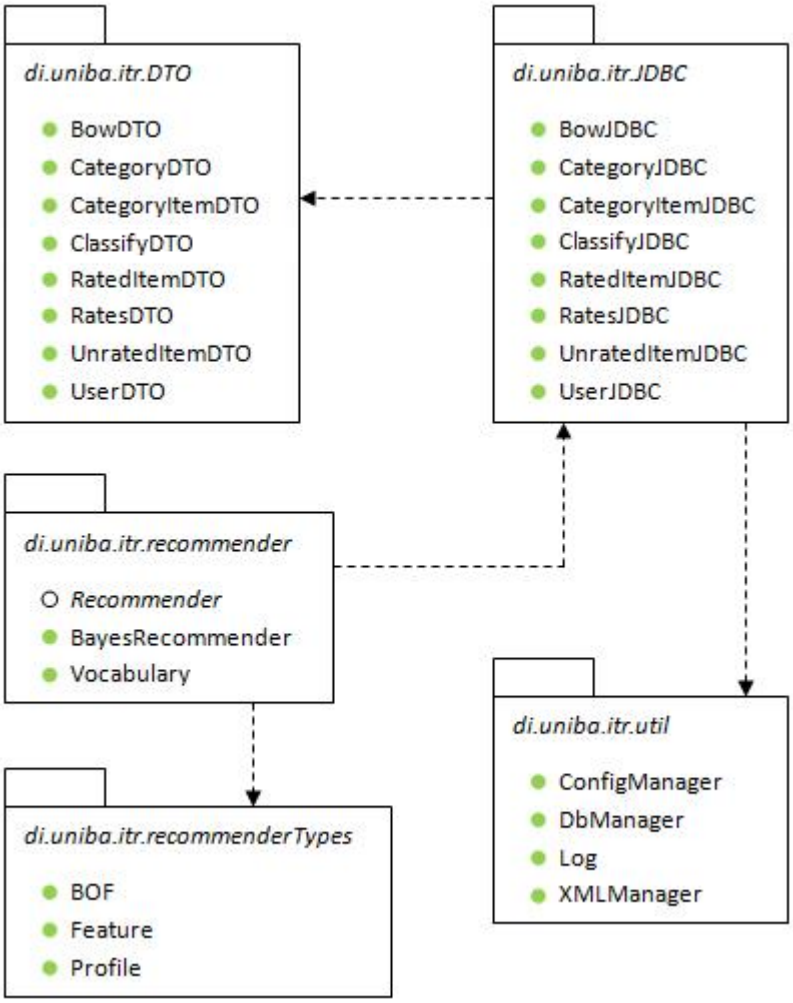

Figura 10 - I package del nuovo ITR

Tramite la figura appena mostrata, andiamo a visualizzare la nuova struttura di ITem Recommender.

In primo luogo, è evidente la nuova suddivisione nei due package principali, *DTO* e *JDBC*. Sia il primo che il secondo mappano fedelmente la struttura del database, con l'aggiunta di due strutture, *RatedItem* e *UnratedItem*, utili il primo per il recupero dei voti già dati agli item (*training*), il secondo per la gestione e previsione degli item non votati (*test*).

Il *Recommender* è stato isolato in un package a parte. Questo è stato realizzato tramite un'*interfaccia* che ne definisce i metodi da implementare, mentre in fase realizzativa è stato implementato il recommender *bayesiano*, lasciando appunto, in futuro, la possibilità di crearne dei nuovi.

Gli ultimi due packages, *RecommenderTypes* e *Util*, sono utili il primo come elemento di supporto al Recommender (inglobando, infatti, le classi *BOF*, *Feature*, *Profile*), il secondo per la gestione accessoria di *file di configurazione*, *database*, *log* e *files XML*.

2.7.1 Package DTO

Il package DTO serve, come detto nei paragrafi precedenti, a gestire il passaggio dei dati, mappando le tabelle del database, con l'aggiunta di due strutture per gestire gli item votati e non da un utente.

BowDTO gestisce gli elementi di tipo Bow del database (1/2)

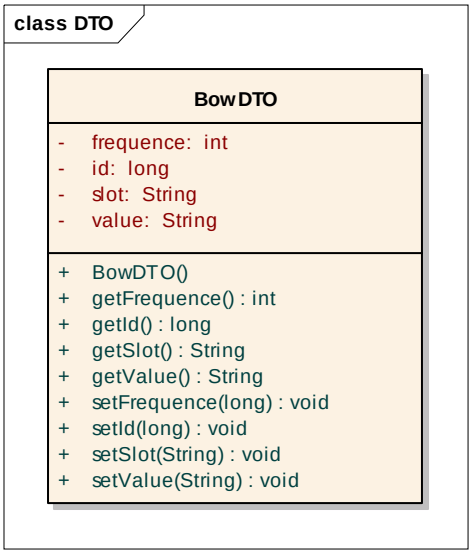

Figura 11 - Classe BowDTO

Metodi (1/2)

Nome	Descrizione
BowDTO()	Costruttore
getFrequence(): int	Restituisce la frequence della BOW
getId(): long	Restituisce l'ID della BOW
getSlot(): String	Restituisce lo slot della BOW
getValue: String	Restituisce il valore associato allo slot della BOW
setFrequence(long): void	Setta la frequence della BOW col valore dato in input

Tabella 11 - Metodi della classe BowDTO (1/2)

BowDTO gestisce gli elementi di tipo Bow del database (2/2)

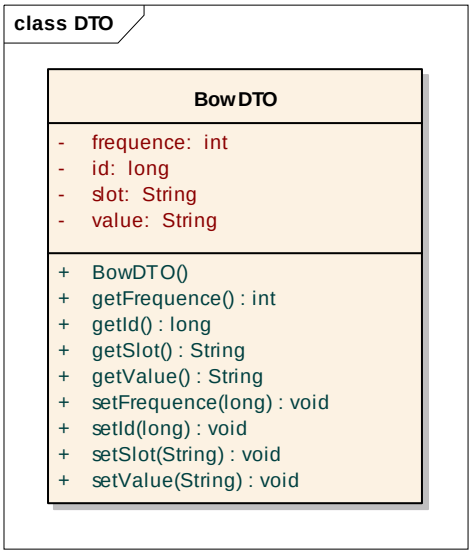

Figura 12 - Classe BowDTO

Metodi (2/2)

Nome	Descrizione
setId(long): void	Setta l'ID della BOW col valore dato in input
setSlot(String): void	Setta lo slot della BOW col valore dato in input
setValue(String): void	Setta il valore dello slot col valore dato in input

Tabella 12 - Metodi della classe BowDTO (2/2)

CategoryDTO gestisce gli elementi di tipo Category del database

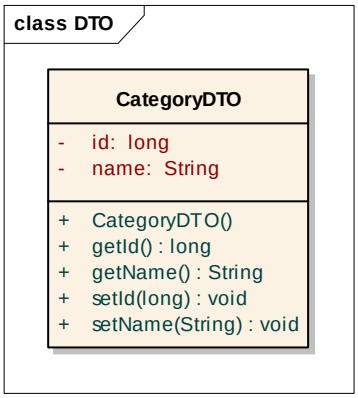

Figura 13 - CategoryDTO

Metodi

Nome	Descrizione
CategoryDTO()	Costruttore
getId(): long	Restituisce l'ID della Category
getName(): String	Restituisce il nome della Category
setId(long): void	Setta l'ID della Category col valore dato in input
setName(String): void	Setta il nome della Category col valore dato in input

Tabella 13 - Metodi della classe CategoryDTO

CategoryItemDTO gestisce gli elementi di tipo CategoryItem del database

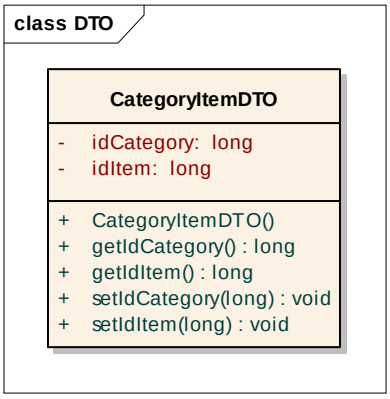

Figura 14 - CategoryItemDTO

Metodi

Nome	Descrizione
CategoryItemDTO()	Costruttore
getIdCategory(): long	Restituisce l'ID della Category
getIdItem(): String	Restituisce l'ID dell'Item
setIdCategory(long): void	Setta l'ID della Category col valore dato in input
setIdItem(String): void	Setta l'ID dell'Item col valore dato in input

Tabella 14 - Metodi della classe CategoryItemDTO

ClassifyDTO gestisce gli elementi di tipo Classify del database (1/2)

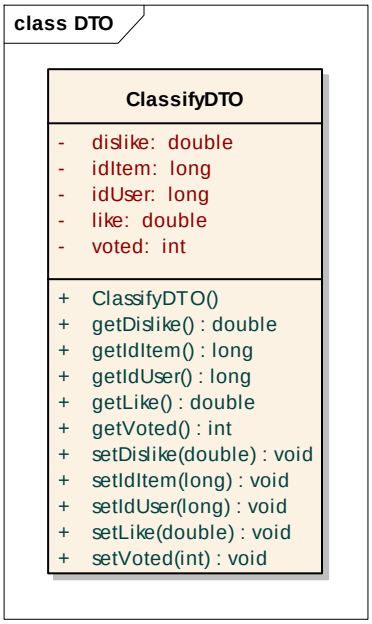

Figura 15 - ClassifyDTO

Metodi (1/2)

Nome	Descrizione
ClassifyDTO()	Costruttore
getDislike(): double	Restituisce il valore di dislike dell'item
getIdItem(): long	Restituisce l'ID dell'item
getIdUser(): long	Restituisce l'ID dell'utente
getLike(): double	Restituisce il valore di like dell'item
getVoted(): int	Restituisce il valore che indica se l'item è stato votato o meno
setDislike(double): void	Setta lo score di dislike col valore dato in input

Tabella 15 - Metodi della classe ClassifyDTO (1/2)

ClassifyDTO gestisce gli elementi di tipo Classify del database (2/2)

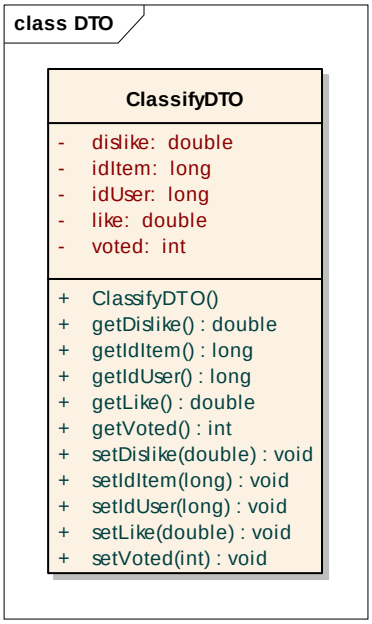

Figura 16 - ClassifyDTO

Metodi (2/2)

Nome	Descrizione
setIdItem(long): void	Setta l'ID dell'item col valore dato in input
setIdUser(long): void	Setta l'ID dell'utente col valore dato in input
setLike(double): void	Setta lo score di like col valore dato in input
setVoted(int): void	Setta il dato che indica se l'item è stato votato o meno col valore dato in input

Tabella 16 - Metodi della classe ClassifyDTO (2/2)

ItemDTO gestisce gli elementi di tipo Item del database (1/2)

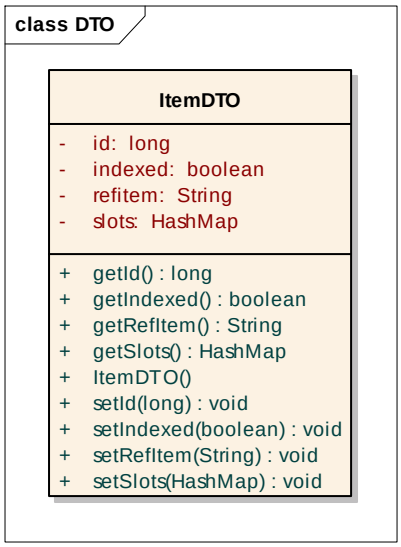

Figura 17 - ItemDTO

Metodi (1/2)

Nome	Descrizione
ItemDTO()	Costruttore
getId(): long	Restituisce l'ID dell'item
getIndexed(): Boolean	Restituisce il booleano che indica se l'item è stato indicizzato
getRefItem(): String	Restituisce il riferimento esterno all'item
getSlots(): HashMap	Restituisce l'HashMap degli slots (dinamici) dell'item
setId(long): void	Setta l'ID dell'item col valore dato in input

Tabella 17 - Metodi della classe ItemDTO (1/2)

ItemDTO gestisce gli elementi di tipo Item del database (2/2)

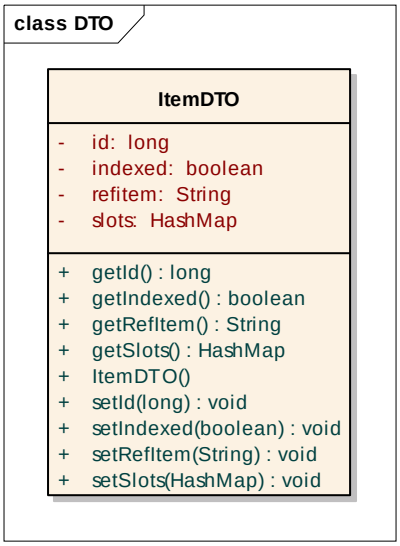

Figura 18 - ItemDTO

Metodi (2/2)

Nome	Descrizione
setIndexed(boolean): void	Setta il booleano di indicizzazione col valore dato in input
setRefItem(String): void	Setta il riferimento esterno all'item col valore dato in input
setSlots(HashMap): void	Setta l'HashMap degli slots con quella data in input

Tabella 18 - Metodi della classe ItemDTO (2/2)

RatedItemDTO gestisce gli elementi di tipo Item votati da un utente (1/2)

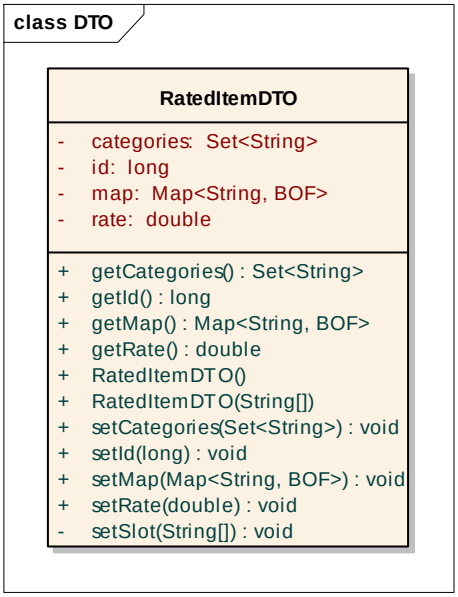

Figura 19 - RatedItemDTO

Metodi (1/2)

Nome	Descrizione
RatedItemDTO()	Costruttore
RatedItemDTO (String[])	Costruttore che inizializza gli slot coi nomi inseriti nell'array di String
getCategories(): Set<String>	Restituisce le categorie dell'item
getId(): long	Restituisce l'ID dell'item
getMap(): Map<String, BOF>	Restituisce la HashMap che rappresenta la bag of features per ciascuno slot
getRate(): double	Restituisce il rate associato all'item
setCategories(Set <String>): void	Setta le categorie dell'item col valore dato in input

Tabella 19 - Metodi della classe RatedItemDTO (1/2)

RatedItemDTO gestisce gli elementi di tipo Item votati da un utente (2/2)

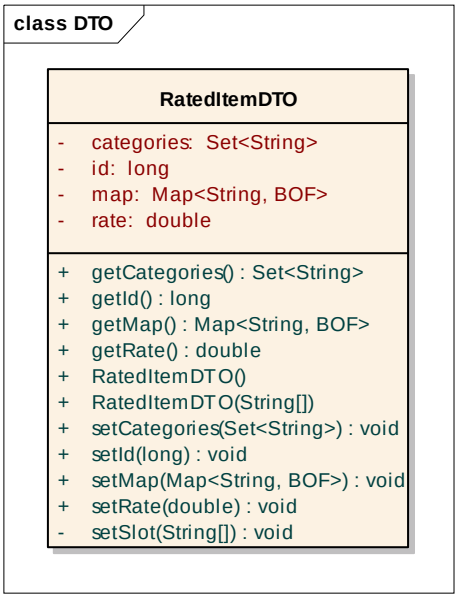

Figura 20 - RatedItemDTO

Metodi (2/2)

Nome	Descrizione
setId(long): void	Setta l'ID dell'item col valore dato in input
setMap(Map<String, BOF>): void	Setta l'HashMap che rappresenta la bag of features per ciascuno slot col valore dato in input
setRate(double): void	Sett ail rate associate all'item col valore dato in input
setSlot(String[]): void	Setta i nomi degli slot del RatedItem col valore dato in input

Tabella 20 - Metodi della classe RatedItemDTO (2/2)

RatesDTO gestisce gli elementi di tipo Rates del database

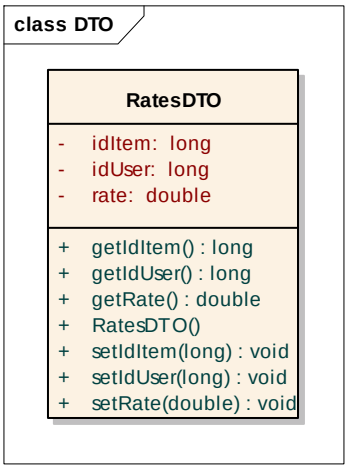

Figura 21 – RatesDTO

Metodi

Nome	Descrizione
RatesDTO()	Costruttore
getIdUser(): long	Restituisce l'ID dell'utente
getIdItem(): long	Restituisce l'ID dell'item
getRate(): double	Restituisce il voto
setIdItem(long): void	Setta l'ID dell'item col valore dato in input
setIdUser(long): void	Setta l'ID dell'utente col valore dato in input
setRate(double): void	Setta il voto col valore dato in input

Tabella 21 - Metodi della classe RatesDTO

UnratedItemDTO gestisce gli elementi di tipo Item non votati da un utente

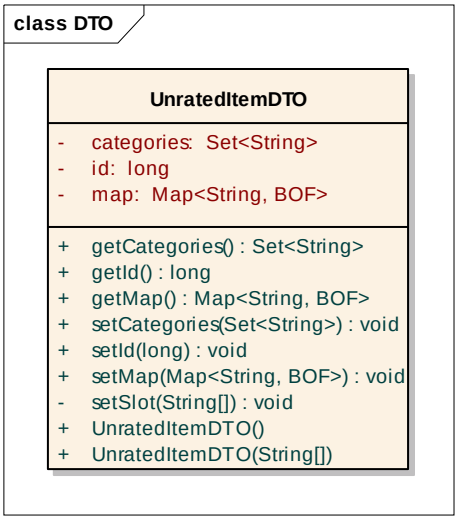

Figura 22 - UnratedItemDTO

Metodi

Nome	Descrizione
UnratedItemDTO()	Costruttore
UnratedItemDTO (String[])	Costruttore che inizializza gli slot coi nomi inseriti nell'array di String
getCategories(): Set<String>	Restituisce le categorie dell'item
getId(): long	Restituisce l'ID dell'item
getMap(): Map<String, BOF>	Restituisce la HashMap che rappresenta la bag of features per ciascuno slot
setCategories(Set <String>): void	Setta le categorie dell'item col valore dato in input
setId(long): void	Setta l'ID dell'item col valore dato in input
setMap(Map<String, BOF>): void	Setta l'HashMap che rappresenta la bag of features per ciascuno slot col valore dato in input
setSlot(String[]): void	Setta i nomi degli slot del RatedItem col valore dato in input

Tabella 22 - Metodi della classe UnratedItemDTO

UserDTO gestisce gli elementi di tipo User del database

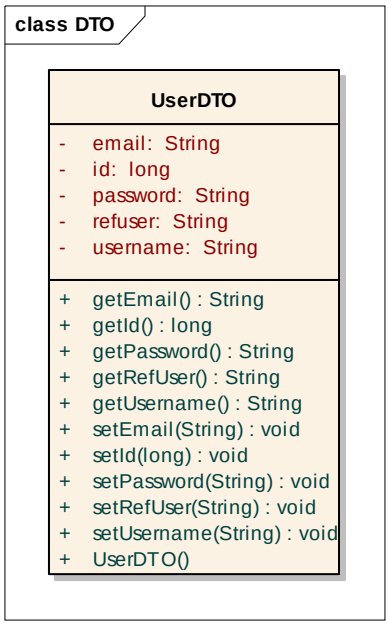

Figura 23 - UserDTO

Metodi

Nome	Descrizione
UserDTO()	Costruttore
getEmail(): String	Restituisce l'email dell'utente
getId(): long	Restituisce l'ID dell'utente
getPassword(): String	Restituisce la password dell'utente
getRefUser(): String	Restituisce il riferimento esterno dell'utente
getUsername(): String	Restituisce l'username dell'utente
setEmail(String): void	Setta l'email dell'utente col valore dato in input
setId(long): void	Setta l'ID dell'utente col valore dato in input
setPassword(String): void	Setta la password dell'utente col valore dato in input
setRefUser(String): void	Setta il riferimento esterno dell'utente col valore dato in input
setUsername(String): void	Setta la username dell'utente col valore dato in input

Tabella 23 - Metodi della classe UserDTO

2.7.2 Package JDBC

Il package JDBC serve, come detto nei paragrafi precedenti, a gestire le operazioni CRUD sul database, mappandone le tabelle, con l'aggiunta di due strutture per gestire gli item votati e non da un utente.

BowJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Bow

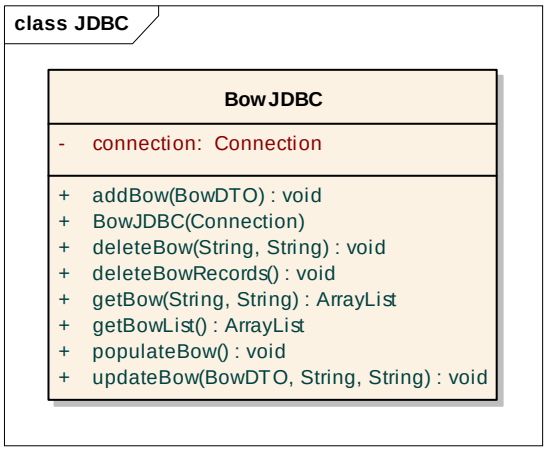

Figura 24 - BowJDBC

Metodi

Nome	Descrizione
BowJDBC (connection)	Costruttore con l'associazione alla connessione al db
addBow(BowDTO): void	Aggiunge una tupla alla tabella Bow
deleteBow(String, String): void	Elimina dalla tabella Bow con condizione field=value
deleteBowRecords(): void	Elimina tutti i record dalla tabella Bow
getBow(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione field=value
getBowList(): ArrayList	Ritorna tutte le tuple di Bow
populateBow(): void	Popola la tabella Bow estrapolando le Bag of Words dagli item
updateBow(BowDTO, String, String): void	Aggiorna le tuple di Bow che soddisfano la condizione field=value

Tabella 24 - Metodi della classe BowJDBC

CategoryItemJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella CategoryItemJDBC

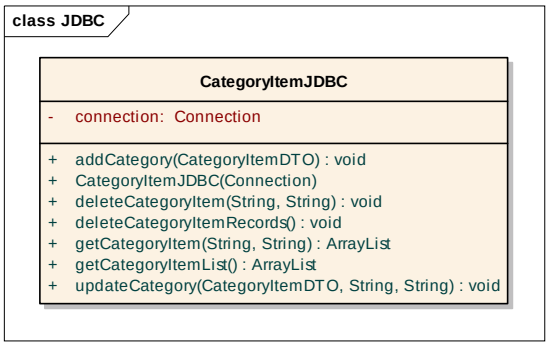

Figura 25 - CategoryItemJDBC

Metodi

Nome	Descrizione
CategoryItemJDBC (connection)	Costruttore con l'associazione alla connessione al db
addCategory (CategoryItemDTO): void	Aggiunge una tupla alla tabella CategoryItem
deleteCategoryItem (String, String): void	Elimina dalla tabella CategoryItem con condizione field=value
deleteCategoryItem Records(): void	Elimina tutti i record dalla tabella CategoryItem
getCategoryItem (String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione field=value
getCategoryItem List(): ArrayList	Ritorna tutte le tuple di CategoryItem
updateCategoryItem (CategoryItemDTO, String, String): void	Aggiorna le tuple di CategoryItem che soddisfano la condizione field=value

Tabella 25 - Metodi della classe CategoryItemJDBC

CategoryJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Category

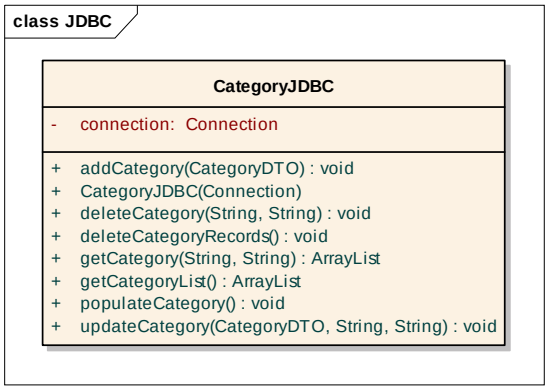

Figura 26 - CategoryJDBC

Metodi

Nome	Descrizione
CategoryJDBC (connection)	Costruttore con l'associazione alla connessione al db
addCategory (CategoryDTO): void	Aggiunge una tupla alla tabella Category
deleteCategory(String, String): void	Elimina dalla tabella Category con condizione field=value
deleteCategory Records(): void	Elimina tutti i record dalla tabella Category
getCategory(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione field=value
getCategoryList(): ArrayList	Ritorna tutte le tuple di Category
populateCategory(): void	Popola la tabella Category tramite i campi salvati nel file di configurazione
updateCategory (CategoryDTO, String, String): void	Aggiorna le tuple di Category che soddisfano la condizione field=value

Tabella 26 - Metodi della classe CategoryJDBC

ClassifyJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Classify

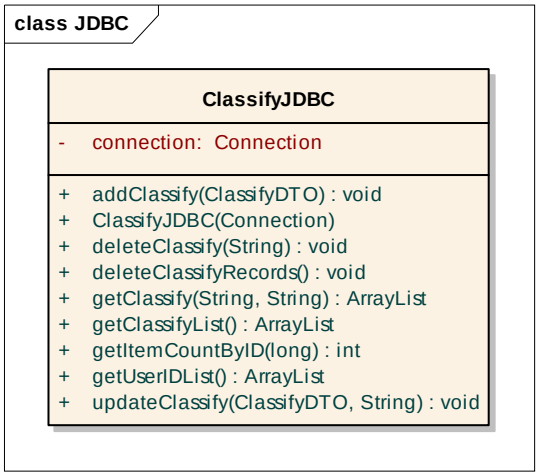

Figura 27 - ClassifyJDBC

Metodi

Nome	Descrizione
ClassifyJDBC (connection)	Costruttore con l'associazione alla connessione al db
addClassify (ClassifyDTO): void	Aggiunge una tupla alla tabella Classify
deleteClassify (String, String): void	Elimina dalla tabella Classify con condizione field=value
deleteClassify Records(): void	Elimina tutti i record dalla tabella Classify
getClassify(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione e l'ordinamento in input
getClassifyList(): ArrayList	Ritorna tutte le tuple di Classify
getItemCountByID (long): int	Ritorna il numero di utenti che sono stati classificati
getUserIDList(): ArrayList	Ritorna la lista degli ID degli utenti già classificati
updateClassify (ClassifyDTO, String): void	Aggiorna le tuple di Bow che soddisfano la condizione in input

Tabella 27 - Metodi della classe ClassifyJDBC

ItemJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Item

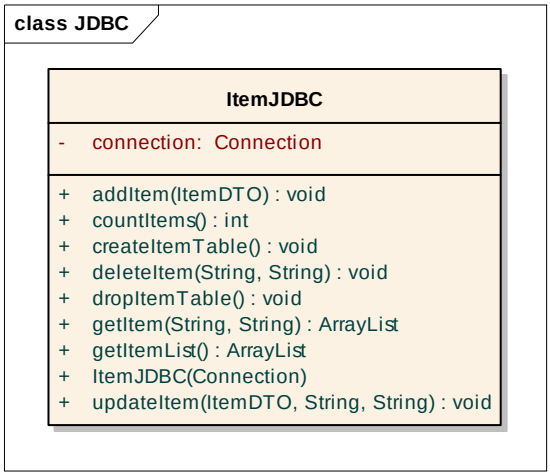

Figura 28 - ItemJDBC

Metodi

Nome	Descrizione
ItemJDBC (connection)	Costruttore con l'associazione alla connessione al db
addItem (ItemDTO): void	Aggiunge una tupla alla tabella Item
countItems(): int	Conta gli item presenti nel db
createItemTable(): void	Crea la tabella item con gli slot salvati nel file di configurazione
deleteClassify (String, String): void	Elimina dalla tabella Item con condizione field=value
dropItemTable(): void	Elimina fisicamente la tabella Item dal db
getItem(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione e l'ordinamento in input
getItemList(): ArrayList	Ritorna tutte le tuple di Item
updateItem (ItemDTO, String, String): void	Aggiorna le tuple di Item che soddisfano la condizione field=value

Tabella 28 - Metodi della classe ItemJDBC

RatedItemJDBC permette la connessione al database e la gestione degli Item votati

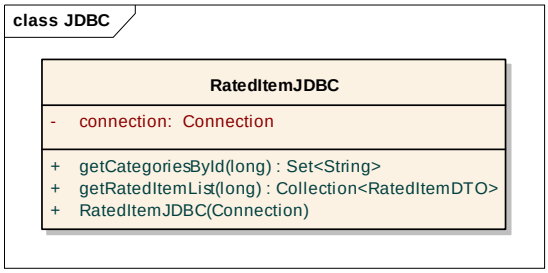

Figura 29 - RatedItemJDBC

Metodi

Nome	Descrizione
RatedItemJDBC (connection)	Costruttore con l'associazione alla connessione al db
getCategoriesById (long): Set<String>	Ritorna le categorie associate all'item con ID dato in input
getRatedItemList (long): Collection <RatedItemDTO>	Ritorna una collezione di item votati dall'utente con ID dato in input

Tabella 29 - Metodi della classe RatedItemJDBC

RatesJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Rates (1/2)

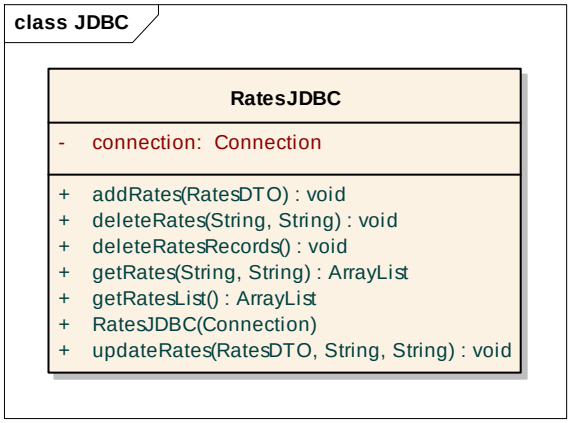

Figura 30 - RatesJDBC

Metodi (1/2)

Nome	Descrizione
RatesJDBC (connection)	Costruttore con l'associazione alla connessione al db
addRates(RatesDTO): void	Aggiunge una tupla alla tabella Rates
deleteRates(String, String): void	Elimina dalla tabella Rates con condizione field=value
deleteRatesRecords(): void	Elimina tutti i record dalla tabella Rates
getRates(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione field=value

Tabella 30 - Metodi della classe RatesJDBC (1/2)

RatesJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Rates (2/2)

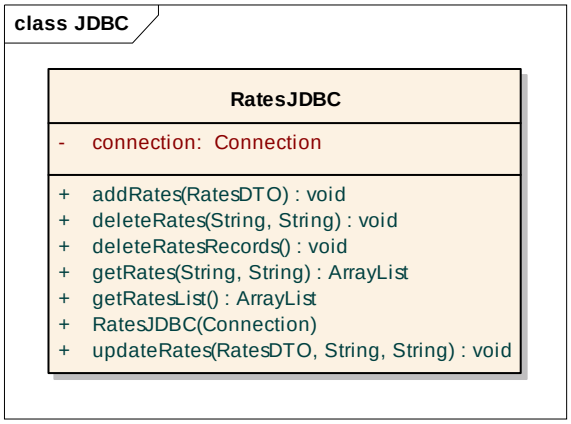

Figura 31 - RatesJDBC

Metodi (2/2)

Nome	Descrizione
getRatesList(): ArrayList	Ritorna tutte le tuple di Rates
updateRates (RatesDTO, String, String): void	Aggiorna le tuple di Rates che soddisfano la condizione field=value

Tabella 31 - Metodi della classe RatesJDBC (2/2)

UnratedItemJDBC permette la connessione al database e la gestione degli Item non votati

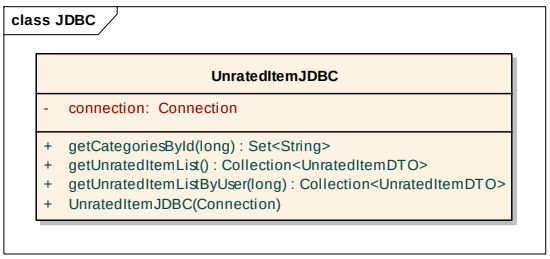

Figura 32 - UnratedItemJDBC

Metodi

Nome	Descrizione
UnratedItemJDBC (connection)	Costruttore con l'associazione alla connessione al db
getCategoriesById (long): Set<String>	Ritorna l'insieme delle categorie associate all'item con ID dato in input
getUnratedItemList(): Collection <UnratedItemDTO>	Ritorna una collezione di tutti gli item non votati da nessuno
getUnratedItemList ByUser(long): Collection <UnratedItemDTO>	Ritorna una collezione di tutti gli item non votati dall'utente con ID dato in input

Tabella 32 - Metodi della classe UnratedItemJDBC

UserJDBC permette la connessione al database e la gestione delle opzioni CRUD sulla tabella Rates

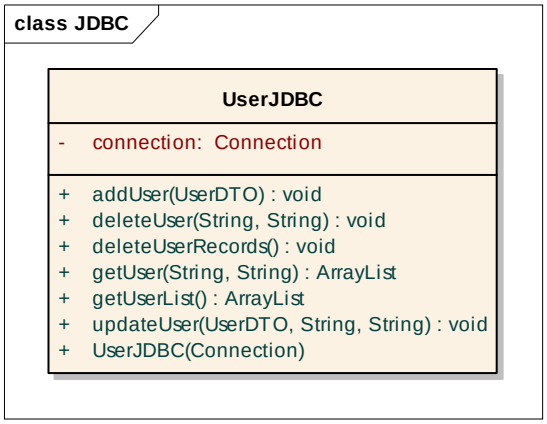

Figura 33 - UserJDBC

Metodi

Nome	Descrizione
UserJDBC (connection)	Costruttore con l'associazione alla connessione al db
addUser(UserDTO): void	Aggiunge una tupla alla tabella User
deleteUser(String, String): void	Elimina dalla tabella User con condizione field=value
deleteUserRecords(): void	Elimina tutti i record dalla tabella User
getUser(String, String): ArrayList	Ritorna un'ArrayList di tuple che soddisfano la condizione field=value
getUserList(): ArrayList	Ritorna tutte le tuple di User
updateUser (UserDTO, String, String): void	Aggiorna le tuple di User che soddisfano la condizione field=value

Tabella 33 - Metodi della classe UserJDBC

2.7.3 Package recommender

Il package recommender contiene l'interfaccia e l'implementazione (Bayesiana) del recommender stesso, più una classe di supporto allo stesso (*Vocabulary*).

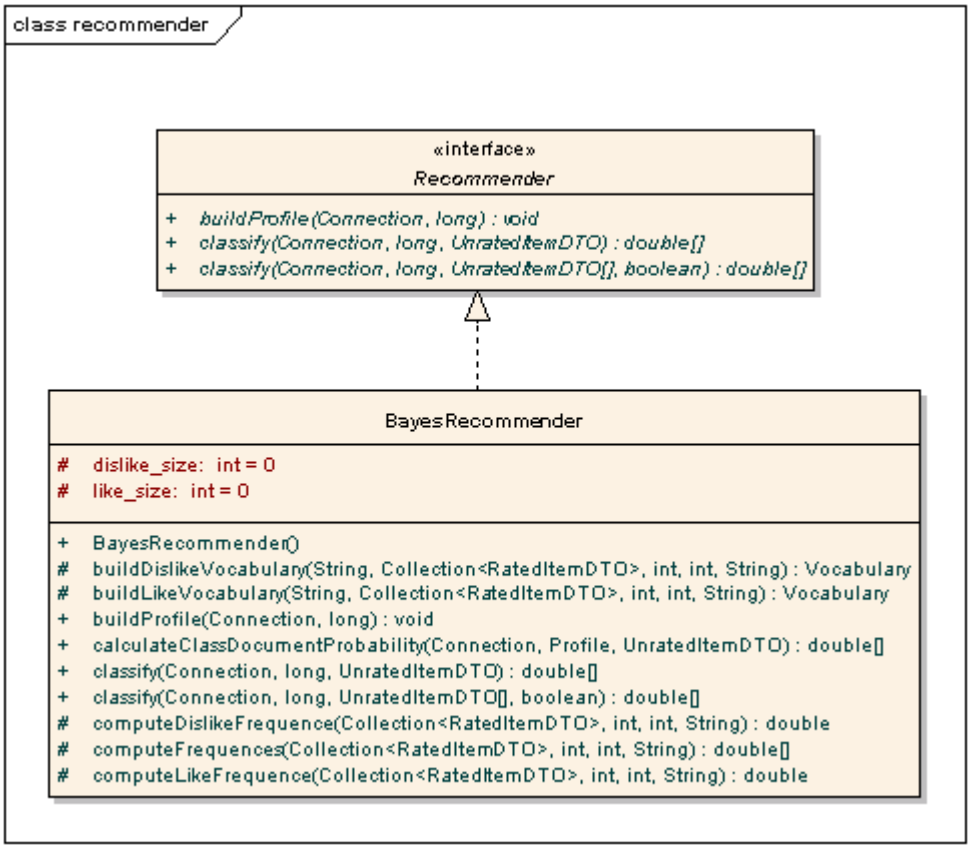

Figura 34 - BayesRecommender + interfaccia Recommender

Metodi (1/3)

Nome	Descrizione
BayesRecommender()	Costruttore
buildDislikeVocabulary (String, Collection<RatedItemDTO, int, int, String): Vocabulary	Ritorna il Vocabolario dislike associato a slot e categoria dati in input, avendo come soglia min e max di voto i due valori in input
buildLikeVocabulary (String, Collection<RatedItemDTO, int, int, String): Vocabulary	Ritorna il Vocabolario like associato a slot e categoria dati in input, avendo come soglia min e max di voto i due valori in input
buildProfile(Connection, long): void	Costruisce il Profilo per l'utente con ID dato in input

Tabella 34 - Metodi della classe BayesRecommender (1/3)

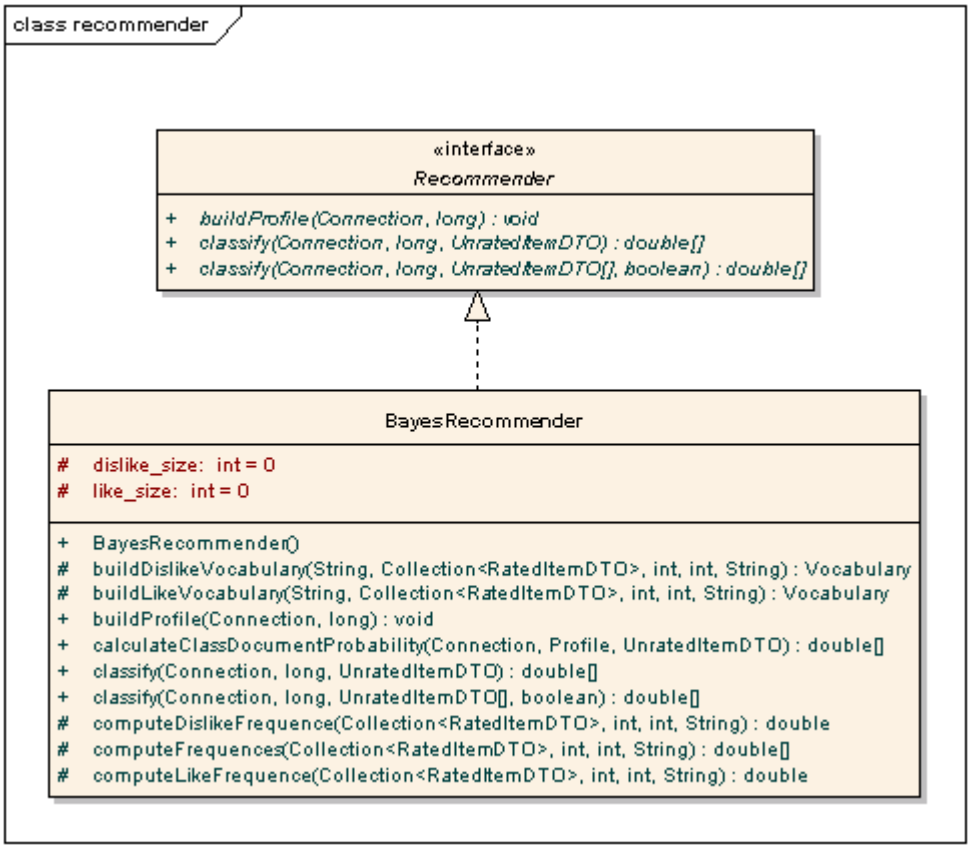

Figura 35 - BayesRecommender + interfaccia Recommender

Metodi (2/3)

Nome	Descrizione
calculateClassDocument Probability(Connection, Profile, UnratedItemDTO): double[]	Calcola la probabilità class-document, dato un profilo ed un item non votato, ritornando un array di due valori, score like e dislike
classify(Connection, long, UnratedItemDTO): double[]	Calcolo dei singoli valori di classify per un item non votato e per l'utente con ID dato in input
classify(Connection, long, UnratedItemDTO[], boolean): double[]	Calcolo della classificazione per tutti gli item non votati dall'utente con ID dato in input. Il valore booleano serve ad indicare se salvare o meno nella tabella Classify anche gli score degli item votati

Tabella 35 - Metodi della classe BayesRecommender (2/3)

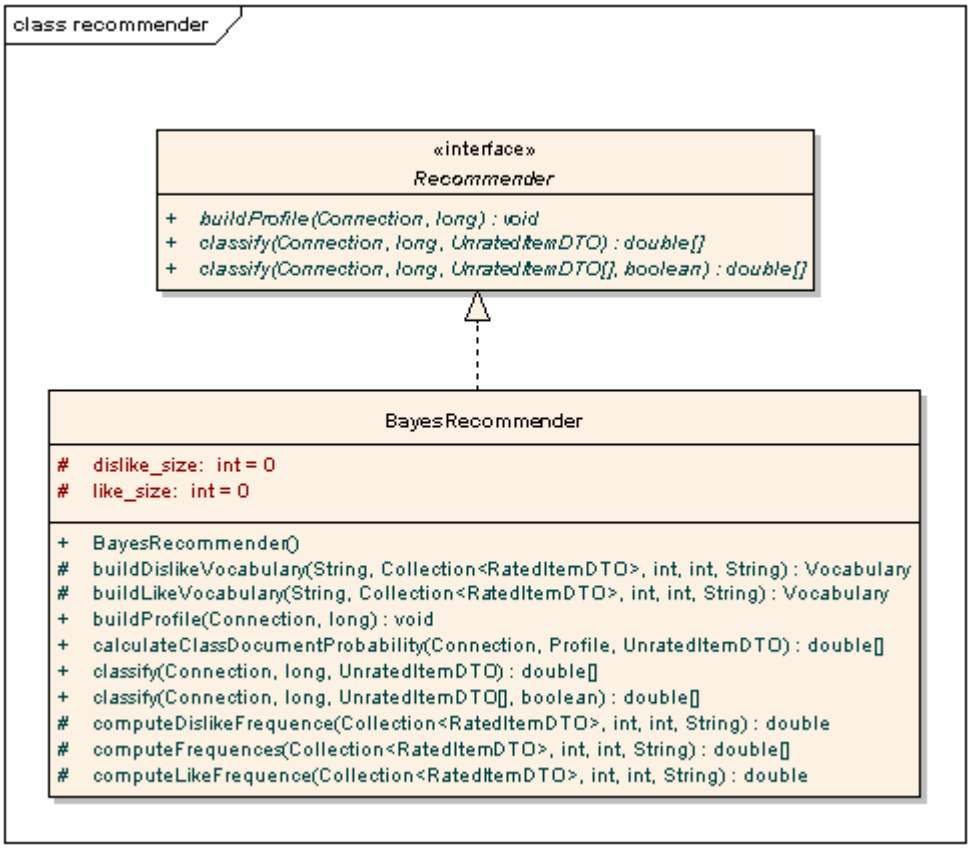

Figura 36 - BayesRecommender + interfaccia Recommender

Metodi (3/3)

Nome	Descrizione
computeDislikeFrequence (Collection<RatedItemDTO>, int, int, String): double	Calcola la dislike frequence, data una collezione di item votati , le soglie min e max, e la categoria date in input
computeFrequences (Collection<RatedItemDTO>, int, int, String): double[]	Calcola le frequences like e dislike, data una collezione di item votati , le soglie min e max, e la categoria date in input
computeLikeFrequence (Collection<RatedItemDTO>, int, int, String): double	Calcola la like frequence, data una collezione di item votati , le soglie min e max, e la categoria date in input

Tabella 36 - Metodi della classe BayesRecommender (3/3)

Vocabulary implementa il Vocabolario utilizzato nel Recommender

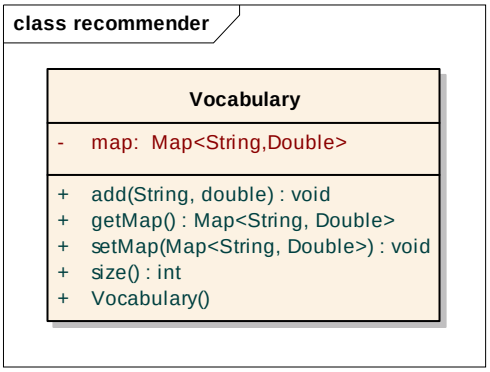

Figura 37 - Vocabulary

Metodi

Nome	Descrizione
Vocabulary()	Costruttore
add(String, double): void	Aggiunge una coppia (valore, frequenza) al vocabolario
getMap(): Map<String, Double>	Ritorna la HashMap contenente tutto il vocabolario
setMap (Map<String, Double>): void	Setta la HashMap (vocabolario) con quella data in input
size(): int	Ritorna la dimensione del vocabolario

Tabella 37 - Metodi della classe Vocabulary

2.7.4 Package recommenderTypes

Il package recommenderTypes funge da supporto al package recommender. Difatti, in esso sono presenti strutture dati utilizzate per la creazione delle raccomandazioni.

BOF implementa la Bag Of Features, un insieme di Features (Valore, Frequenza)

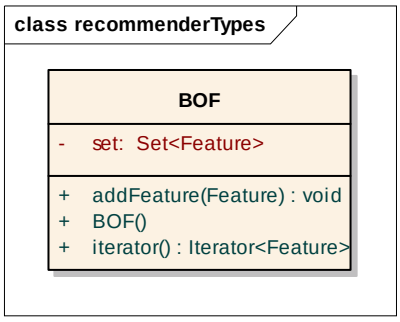

Figura 38 - BOF

Metodi

Nome	Descrizione
BOF()	Costruttore
addFeature (Feature): void	Aggiunge una nuova Feature alla Bag of Features
iterator(): Iterator<Feature>	Ritorna l'iteratore della Bag of Features

Tabella 38 - Metodi della classe BOF

Feature implementa le coppie <Valore, Frequenza> in una struttura dati

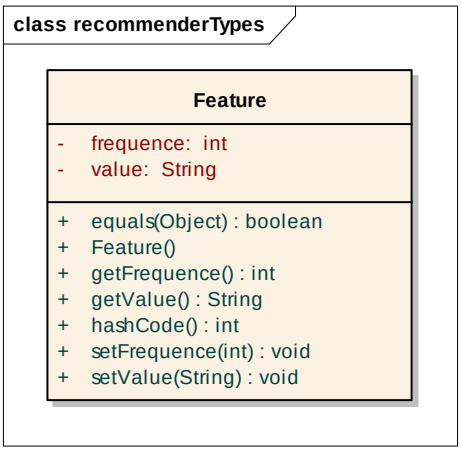

Figura 39 - Feature

Metodi

Nome	Descrizione
Feature()	Costruttore
equals(Object): boolean	Restituisce vero se due features sono uguali, falso altrimenti
getFrequence(): int	Ritorna la frequence della feature
getValue(): String	Ritorna il valore della feature
hashCode(): int	Ritorna il codice Hash associato alla feature
setFrequence(int): void	Setta la frequence con in valore dato in input
setValue(String): void	Setta il valore con quello dato in input

Tabella 39 - Metodi della classe Feature

Profile è la classe incaricata di creare, caricare, gestire e salvare i profili in formato XML (1/2)

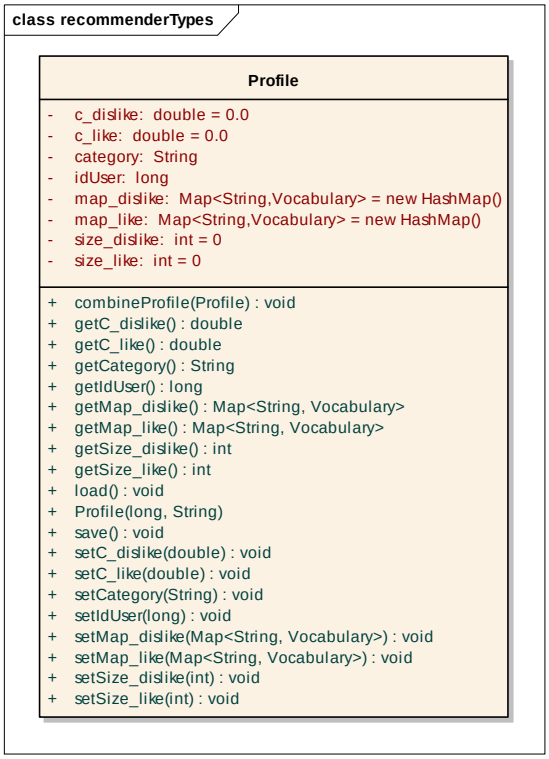

Figura 40 - Profile

Metodi (1/2)

Nome	Descrizione
Profile(long, String)	Crea un nuovo profilo con ID utente e categoria dati in input
combineProfile (Profile): void	Combina due profili in uno
getC_dislike(): double	Ritorna c_like
getC_like(): double	Ritorna c_dislike
getCategory(): String	Ritorna la categoria del profilo
getIdUser(): long	Ritorna l'ID utente del profilo
getMap_dislike(): Map<String, Vocabulary>	Ritorna la HashMap relativa a dislike
getMap_like(): Map<String, Vocabulary>	Ritorna la HashMap relativa a like
getSize_dislike(): int	Ritorna size_dislike
getSize_like(): int	Ritorna size_like
load(): void	Carica un profilo da file XML in un'istanza di Profile
save(): void	Salva un profilo da istanza di Profile a file XML
setC_dislike(double): void	Setta c_like con il valore dato in input
setC_like(double): void	Setta c_dislike con il valore dato in input
setCategory(String): void	Setta la categoria del profilo con il valore dato in input

Tabella 40 - Metodi della classe Profile (1/2)

Profile è la classe incaricata di creare, caricare, gestire e salvare i profili in formato XML (2/2)

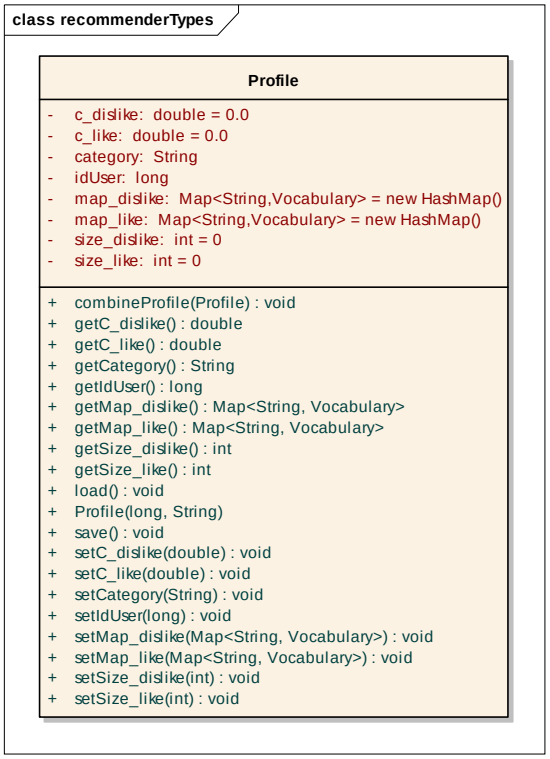

Figura 41 - Profile

Metodi (2/2)

Nome	Descrizione
setIdUser(long): void	Setta l'ID utente del profilo con il valore dato in input
setMap_dislike (Map<String, Vocabulary>): void	Setta la HashMap relativa a dislike con quella data in input
setMap_like (Map<String, Vocabulary>): void	Setta la HashMap relativa a like con quella data in input
setSize_dislike(int): void	Setta size_dislike con il valore dato in input
setSize_like(int): void	Setta size_dislike con il valore dato in input

Tabella 41 - Metodi della classe Profile (2/2)

2.7.5 Package util

Il package util contiene tutte quelle classi che si occupano di *gestione del file di configurazione, connessione al database, gestione del Log* e *lettura/scrittura del file XML relativi ai profili.*

ConfigManager si occupa della gestione del file di configurazione di ITR

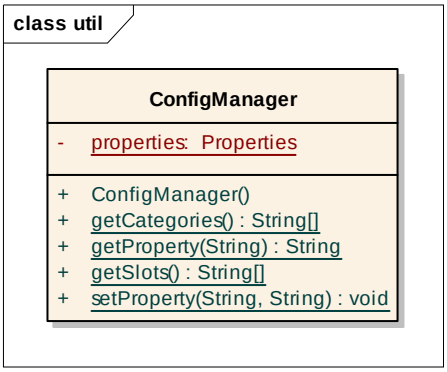

Figura 42 - ConfigManager

Metodi

Nome	Descrizione
ConfigManager()	Costruttore
getCategories(): String[]	Ritorna le categorie salvate nel file di configurazione
getProperty(String): String	Ritorna il valore della proprietà data in input
getSlots(): String[]	Ritorna gli slots salvati nel file di configurazione
setProperty(String, String): void	Setta, coi valori dati in input, la proprietà ed il valore corrispondente.

Tabella 42 - Metodi della classe ConfigManager

DbManager si occupa di creare e ritornare la connessione al database

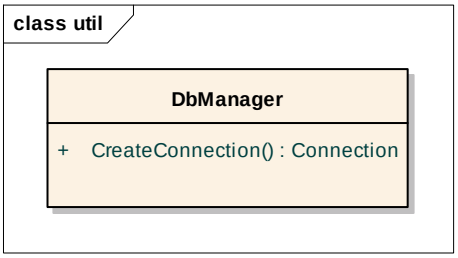

Figura 43 - DbManager

Metodi

Nome	Descrizione
CreateConnection(): Connection	Crea e ritorna la connessione al database tramite i parametri recuperati dal file di configurazione

Tabella 43 - Metodi della classe DbManager

Log si occupa della gestione del file di Log di ITR

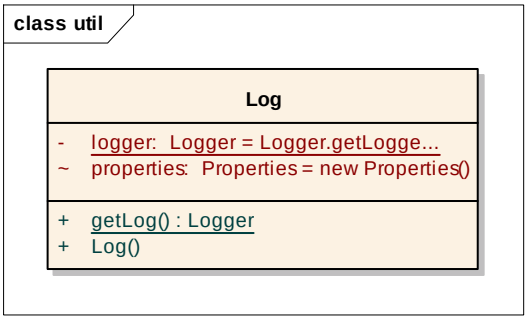

Figura 44 - Log

Metodi

Nome	Descrizione
Log()	Costruttore
getLog(): Logger	Ritorna il Log di ITR

Tabella 44 - Metodi della classe Log

XMLManager si occupa della gestione dei file XML relativi ai profili degli utenti di ITR (1/2)

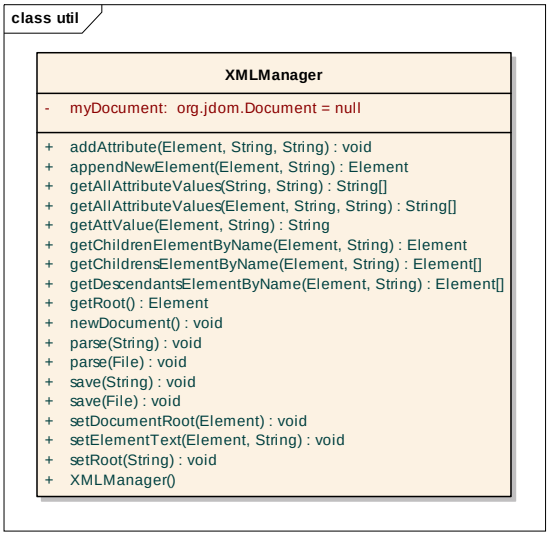

Figura 45 - XMLManager

Metodi (1/2)

Nome	Descrizione
XMLManager()	Costruttore
addAttribute(Element, String, String): void	Aggiunge un attributo all'elemento del file XML
appendNewElement (Element, String): Element	Aggiunge un figlio all'elemento del file XML
getAllAttributeValues (String, String): String[]	Ritorna tutti i valori degli attribute del file XML
getAllAttributeValues (Element, String, String): String[]	Ritorna tutti I valori dell'attributo del file XML con nome dato in input
gettAttValue(Element, String): String	Ritorna il valore dell'attributo di un dato elemento in input
getChildrenElement ByName(Element, String): Element	Ritorna il primo figlio dell'elemento dato in input
getChildrensElement ByName(Element, String): Element[]	Ritorna i figli dell'elemento dato in input
getDescendants ElementBy Name(Element, String): Element	Ritorna i discendenti dell'elemento dato in input
getRoot(): Element	Ritorna la radice del file XML
newDocument(): void	Azzera l'istanza del file XML
Parse(String): void	Carica il file XML nel JDOM
Parse(File): void	Carica il file XML nel JDOM

Tabella 45 - Metodi della classe XMLManager (1/2)

XMLManager si occupa della gestione dei file XML relativi ai profili degli utenti di ITR (2/2)

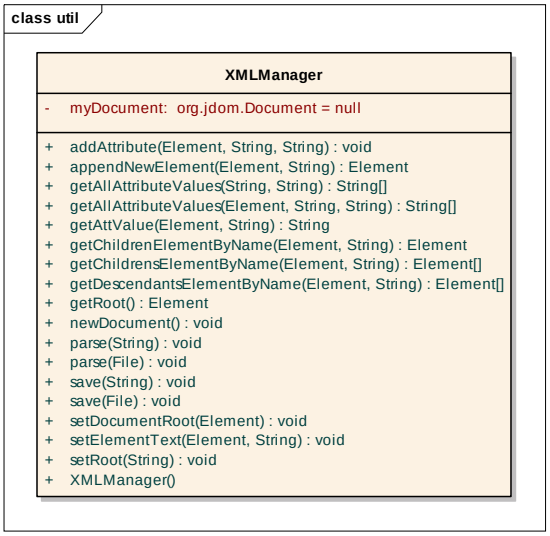

Figura 46 - XMLManager

Metodi (2/2)

Nome	Descrizione
Save(String): void	Salva il JDOM nel file specificato
Save(File): void	Salva il JDOM nel file specificato
SetDocumentRoot (Element): void	Imposta la radice del file XML con l'elemento dato in input
setElementText (Element, String): void	Imposta il testo associato all'elemento con quello dato in input
setRoot(String): void	Imposta la radice del file con un elemento avente come nome quello dato in input

Tabella 46 - Metodi della classe XMLManager (2/2)

3. ITR in uno scenario di accesso ai beni culturali

Obiettivo di questo capitolo è quello di capire in che modo integrare ITem Recommender all'interno di uno scenario di accesso ai beni culturali. Per questo motivo, verrà analizzato in dettaglio il progetto CHAT – "Cultural Heritage fruition & e-learning applications of new Advanced (multimodal) Technologies" nei suoi obiettivi, fino ad arrivare a comprendere in che maniera sfruttare le features di raccomandazione del sistema reingegnerizzato.

3.1 Progetto CHAT

L'obiettivo di CHAT è la costruzione di una piattaforma per la creazione di servizi multimodali mobili. Un utente avente a disposizione un terminale mobile può interagire con esso utilizzando più modi sinergici (due o più input convergono verso la definizione di un singolo comando che una volta eseguito produce uno o più output su modalità diverse). L'obiettivo è di creare una piattaforma adattiva, ossia in grado di acquisire informazioni in modo intelligente e selezionare le modalità in modo opportuno, sia per quanto riguarda gli input da proporre al cliente, sia per quanto riguarda la gestione di essi una volta immessi nel sistema. A maggior ragione la adattività del sistema deve potersi applicare alla gestione del migliore output possibile, a fronte di una richiesta di servizio mediante interfaccia multimodale.

3.2 Panoramica sugli obiettivi di CHAT

Le comunicazioni e le interazioni tra esseri gli umani sono:

- *multimodali*, cioè avvengono coinvolgendo diverse percezioni sensoree (modalità) interpretate contestualmente: mentre parliamo gesticoliamo, mostriamo approvazione o disapprovazione con l'espressione del viso e la postura del corpo, prendiamo e mostriamo appunti fatti di testo e grafici.

- *adattive*, cioè i soggetti coinvolti adeguano reciprocamente il linguaggio, la complessità degli argomenti trattati, le modalità di supporto al linguaggio, onde favorire l'efficacia del processo di comunicazione.

Nella comunicazione umana, l'uso simultaneo di diverse modalità avviene sempre, anche inconsapevolmente. È esperienza comune, ad esempio, il fatto che spesso gesticoliamo anche parlando al telefono.

Per analogia, possiamo definire i sistemi informatici e i servizi telematici come:

- *multimodali* se, mediante idonee tecnologie e una adeguata progettazione dell'interazione uomo-macchina, l'interazione stessa avviene in modo naturale, consentendo, oltre alle modalità classiche delle interfacce grafiche, il riconoscimento di modalità di comunicazione quali il linguaggio parlato, la scrittura corsiva, alcuni gesti, …

- *adattivi* se, grazie ad idonee tecnologie, essi sono in grado di adattare il loro comportamento agli obiettivi, ai compiti, agli interessi, o ad altre definibili caratteristiche, di utenti individuali e di gruppi di utenti.

Il primo sistema dimostrativo multimodale realizzato, il sistema “Put That There”, documentato nel classico lavoro di Bolt del 1980 [24], elaborava comandi vocali in parallelo ad un puntamento di tipo touch-pad, consentendo all’utente di creare e muovere oggetti grafici su un display 2D.

Ad esempio, l’utente poteva impartire il comando *“Create a blue square there”* dove la posizione del “there” era indicata da un cursore su uno schermo 2D.

L’elaborazione semantica era basata sul riconoscimento del parlato, ed il significato del lemma “there” era stabilito individuando le coordinate x, y indicate dal cursore nell’istante in cui “there” era pronunciato.

Da allora sono stati realizzati, sempre in ambito di ricerca, alcuni sistemi sperimentali, anche interessanti, ma che sono sempre piuttosto limitati, nel senso che ciascuno di essi risente di importanti limitazioni relative sia al numero di modalità diverse abilitate, sia al contesto di applicazione, spesso definito in modo piuttosto specifico.

Studi sperimentali condotti su sistemi multimodali hanno confermato miglioramenti sensibili nella facilità e nell’efficacia della interazione. In particolare è stato mostrato come le interfacce multimodali:

- supportano interazioni più vicine al naturale protocollo espressivo umano, con vantaggi sia in termini di efficienza sia di naturalezza;
- rispetto ai servizi basati sul solo riconoscimento vocale, possono migliorare l’affidabilità mediante il riconoscimento integrato del contenuto informativo fornito da diverse modalità, il che è particolarmente utile quando il riconoscimento vocale avviene in contesti ambientali difficili;

- permettendo l’uso di diverse modalità, aumentano il controllo dell’utente sulla interazione, e quindi risultano più facilmente accessibili da utenti svantaggiati quali utenti anziani, utenti che soffrono di handicap percettivi o motori, utenti che non sono madrelingua o anche solo che parlano con forti inflessioni dialettali.

Quest’ultimo aspetto è particolarmente importante per limitare possibili fenomeni di ulteriore emarginazione, sociale ed economica, delle fasce più deboli della popolazione.

Dal canto loro le tecnologie per l’adattività consentono di superare i limiti di un approccio indifferenziato oggi ancora molto diffuso, fornendo tipicamente tre prestazioni fondamentali:

- il supporto adattivo della interazione, ad esempio quando l’utente naviga in un sito web, un sistema adattivo può manipolare i link ordinandoli, annotandoli, o anche nascondendone alcuni, per migliorare la navigazione;

- la selezione adattiva del contenuto, vale a dire, quando l’utente cerca informazioni, il sistema adattivo può stabilire delle priorità, scegliendo quelle più rilevanti;

- la presentazione adattiva, cioè, quando l’utente raggiunge le informazioni o i servizi di cui necessita, il sistema può adattare la presentazione del contenuto alle preferenze dell’utente o ai dispositivi di cui dispone.

La realizzazione di sistemi/servizi in cui coniugare la multimodalità con l’adattività apre la possibilità di interessanti sinergie, quali ad esempio la selezione adattiva delle modalità di interazione o l’uso di informazioni, sull’utente e sul suo contesto, per migliorare la probabilità di un corretto

riconoscimento di significati semantici complessi convogliati da diverse modalità attive simultaneamente.

L'adattività è certamente tra i principali filoni di ricerca sulla multimodalità [25], filone che è ancora poco esplorato a parte alcune ricerche su tecniche adattive per migliorare la robustezza dei riconoscimenti vocali in ambienti rumorosi. [26] [27] [28]

I sistemi/servizi multimodali adattivi promettono vantaggi in particolare ai fini della fruizione mobile di applicazioni e servizi.

Gli utenti mobili, infatti:

- generalmente usano dispositivi *piccoli* e *leggeri*, come telefoni cellulari e PDA, i cui schermi e tastiere, se presenti, sono di dimensioni molto ridotte;
- possono incontrare contesti ambientali difficili, come *ambienti rumorosi* o zone di coperture wireless di minori prestazioni;
- devono relazionarsi con l'ambiente reale, talvolta svolgendo attività delicate, come la guida di un auto.

È facilmente intuibile, quindi, come i servizi mobili possano giovarsi, in termini di naturalezza, efficacia e sicurezza della fruizione, della possibilità di adattare sia le modalità di interazione, sia la presentazione dei contenuti, alle capacità hw/sw del dispositivo in uso, alle situazioni ambientali, ed a specifiche esigenze di uso.

In particolare, la disponibilità di varie modalità di input, come il riconoscimento della voce, di gesti/segni grafici e della scrittura corsiva, è di importanza fondamentale sia ai fini di una mera possibilità di scelta, sia

per permettere la cooperazione di diversi modi di ingresso nella costruzione di richieste complesse.

Sul mercato sono da alcuni anni disponibili, con buoni risultati:

- tecnologie unimodali di riconoscimento e sintesi vocale per l'erogazione automatica di servizi via telefono;
- riconoscitori vocali (IBM Via Voice, Dragon Naturally Speaking), o di scrittura corsiva (MS Windows for Tablet PC), utilizzabili come sostituti della tastiera, nell'ambito della tradizionale interfaccia GUI dei PC.

Non è invece ancora disponibile nessun sistema di valenza generale in grado di interpretare input forniti da diverse modalità attive contemporaneamente. Come detto, sistemi siffatti sono stati realizzati solo in ambiti di ricerca, tipicamente integrando solo due distinte modalità.

Per i servizi telematici su Internet la situazione è analoga: oggi non è ancora possibile realizzare pagine Web multimodali fruibili su PC con comuni browser, tantomeno è possibile realizzare servizi multimodali fruibili mediante dispositivi wireless, quali PDA o Smartphones.

La diffusione su larga scala di servizi multimodali adattivi fruibili anche su dispositivi wireless leggeri, dunque, vede ancora grandi spazi per una ricerca, multidisciplinare, che spazi dalle *metodologie e strumenti di progettazione*, alle *architetture*, a specifici *algoritmi di implementazione*.

La finalità del progetto CHAT è quindi proprio di effettuare studi ed esperienze onde acquisire nuove conoscenze, utili alla realizzazione, in momenti successivi, di nuovi sistemi e servizi multimodali adattivi.

Particolare enfasi è stata data agli studi sulle problematiche connesse alla fruizione di siffatti servizi mediante dispositivi leggeri, standard, e collegati in reti wireless. Tra queste è preminente la verifica dell'usabilità, dal punto di vista degli utenti finali, dei sistemi/servizi multimodali su dispositivi con input/output limitati rispetto ai quali attualmente gli utenti hanno incertezze o perplessità derivanti dalla complessità di interazione.

3.3 Architettura del sistema

Lo schema dell'architettura di CHAT è mostrato in figura:

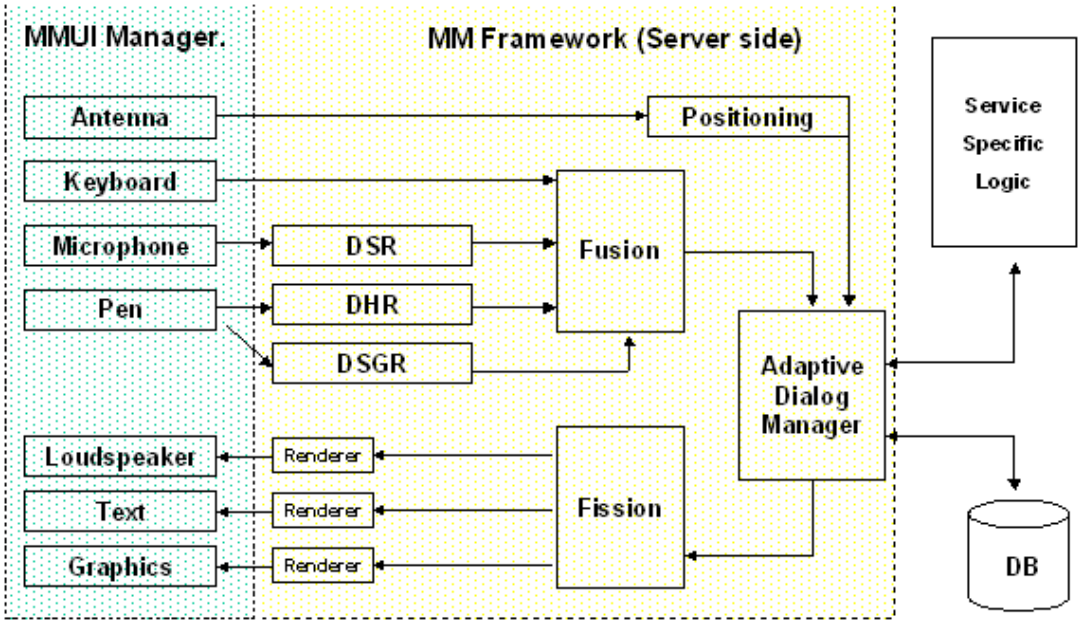

Figura 47 - Architettura del sistema

In sintesi, scorrendo il diagramma da sinistra a destra, notiamo:

- sulla sinistra sono evidenziate le modalità trattate;
- il dispositivo, oltre all'hw/sw per acquisire tali segnali, dispone di un componente software, leggero e basato su standard, che gestisce

l’interfaccia multimodale in tutti gli aspetti non delegabili sulla rete (MMUI Manager);

- il MMUI Manager riconosce i segnali relativi a ciascuna modalità di input, e li trasmette verso opportuni server, gestendo i protocolli e gli standard di comunicazione necessari (approccio thin client);

- sulla rete sono disponibili, in parallelo, riconoscitori del parlato naturale (DSR), della scrittura corsiva (DHR), e di insiemi predefiniti di gesti e simboli grafici (DSGR);

- il sistema deve consentire l’interpretazione contestuale di diversi input (multimodale simultaneo coordinato): definiamo "Fusion" (o meglio “Late Fusion” o “Semantic Fusion”) tale fondamentale funzione;

- per la Semantic Fusion lo stato dell’arte prevede una rappresentazione semantica comune (typed feature structures [29]), una operazione di unificazione (typed feature structures unification) ed infine approcci statistici volti ad identificare il risultato più probabile [30], da fornire in ingresso all’ Adaptive Dialog Manager;

- l’Adaptive Dialog Manager si occupa della gestione generale del dialogo tra sistema ed utente, mantenendo e aggiornando un modello complessivo dell’interazione, al fine di risolvere problemi tipici di un dialogo, quali ad esempio la risoluzione di riferimenti anaforici, riferimenti deittici, ecc;

- l’Adaptive Dialog Manager, inoltre, costruisce opportuni modelli dell’utente e del suo contesto, che sono utilizzati per adattare/ personalizzare l’evoluzione complessiva del dialogo, la resa delle

informazioni in uscita verso l’utente (Fission), gli algoritmi di riconoscimento multimodale (Fusion);

- il modulo di Fission, infine, implementa opportune logiche di distribuzione dell'informazione sui diversi possibili modi di output paralleli, anche ottimizzandone la resa in funzione delle informazioni disponibili sull’utente e sul suo contesto.

Una architettura software di questo tipo, laddove sia possibile utilizzare opportuni standard nel dialogo tra componenti diversi, consente una ottima flessibilità nella sostituzione di moduli e componenti. Questo è molto importante in un contesto di tecnologie in rapida evoluzione. Ad esempio, tecnologie quali la Speech Recognition (DSR) e l'Handwriting Recognition (DHR) sono in continuo progresso.

Una architettura aperta consente quindi la sostituzione di moduli specializzati senza che siano necessarie modifiche alle applicazioni.

Un beneficio collaterale importante di questo approccio modulare è una migliore gestione del rischio tecnico di progetto.

Questa architettura ha caratterizzato il progetto CHAT sin dalla sua prima presentazione. Successivamente alla presentazione del progetto, il W3C ha proposto il Multimodal Interaction Framework. L’architettura appena introdotta è perfettamente compatibile con il Multimodal Interaction Framework, anzi è del tutto simile, a parte banali differenze nei nomi dei componenti.

3.4 Analisi di scenari: accesso a beni culturali

In questo paragrafo analizzeremo due possibili scenari per l'accesso a beni culturali, in linea con quanto previsto all'interno del progetto CHAT:

1. Guida museale;
2. Guida ad un sito archeologico.

3.4.1 Guida museale

Indispensabile per la pianificazione e l'esecuzione dello scenario museale è la scelta della tipologia di museo su cui realizzare un pianificatore. È stata fatta la scelta di una pinacoteca rispetto ad un altro tipo di museo, a causa della maggiore semplicità strutturale di tale tipologia museale rispetto ad altri tipi. Di fatto si tratta di un tipo di museo separato in sale contenenti solo, o almeno prevalentemente, quadri appesi al muro. Si evitano quindi complicazioni quali la presenza di statue, spazi centrali condivisi etc.

Ad un utente, all'ingresso del museo, viene fornito un device (PDA/ smartphone) con un'applicazione già installata ed avviata. Il device in questione deve essere in grado di gestire diversi canali di input. Nello scenario considerato esso deve essere almeno in grado di:

- acquisire audio;
- essere dotato di touch screen (stilo);
- acquisire sketch;
- avere accesso a rete wireless.

L’utente comincia la propria visita entrando nelle diverse stanze del museo. Grazie a dei sensori di localizzazione è possibile sapere in quale stanza del museo egli si trovi. Tramite un collegamento wireless l’applicazione scambia informazioni con i server che rispondono alle richieste dell’utente e a determinati input ambientali.

All’ingresso di ogni sala i contenuti presentati all’utente si adattano spontaneamente mostrando il catalogo delle opere presenti all’interno dell’area. L’utente può selezionare con lo stilo l’opera a cui è interessato e su questa chiedere ulteriori informazioni. In questa fase supponiamo che l’utente selezioni (toccando con lo stilo un punto dell’immagine relativa) un’opera ed utilizzi la voce per specificare contemporaneamente un’ulteriore richiesta di informazione (ad esempio “autore”, “dettagli” , “epoca”, ecc).

L’applicazione deve essere in grado di riconoscere, ricostruire e disambiguare le informazioni disponibili, e presentare all’utente i contenuti richiesti (ad esempio avviando un filmato, facendo partire una spiegazione solo vocale oppure mostrando del testo o delle immagini).

I comandi possono essere dati via voce o via sketch/handwriting (per esempio in ambienti rumorosi, o nel caso in cui l’utente preferisca questa specifica forma di interazione).

L’acquisizione della voce dovrebbe avvenire in maniera continua, ossia deve essere sempre aperto un canale audio tra client e server per trasmettere l’audio acquisito dal terminale. Il riconoscimento dell’audio avverrà a livello del server (distributed recognition).

L‘acquisizione del point, dello sketch o dell’handwriting avviene invece in maniera puntuale, ossia inizia quando lo stilo tocca un oggetto sensibile e termina quando lo stilo si solleva dallo schermo. Quanto acquisito viene

inviato al server che provvederà ad inviare agli specifici riconoscitori (distributed recognition).

È previsto che i comandi vocali non siano parole (o sequenze di parole) univocamente predeterminate, ma che siano estrapolabili dal linguaggio naturale dell'utente. Per esempio il comando "zoom" potrà essere impartito mediante diverse locuzioni: "Ingrandisci", "Ingrandisci la figura", "più grande", "fai lo zoom", ecc. Lo stesso non si può dire per il canale di sketch, per il quale è previsto un determinato insieme di simboli, ciascuno dei quali con un significato specifico.

L'insieme delle informazioni relative al museo nel suo complesso, alle singole sale, alle opere e agli autori sarà contenuto all'interno di una specifica struttura dati. Come repository è stato scelto il DB multimediale *Fedora*.

Il DB è strutturato come una collezione di Digital Objects ciascuno dei quali può contenere diversi tipi di contenuto digitale (datastream):

- immagine (formato jpg);
- audio (wav o mp3);
- video (mpeg);
- testo (txt o anche html).

A ciascun oggetto possono quindi essere associati diversi tipi di contenuto e, per ciascun tipo, diversi contenuti (ad es. testi diversi con un diverso livello di approfondimento).

3.4.2 Guida ad un sito archeologico

La visita ad un parco archeologico permette di considerare un ambiente outdoor e di affrontare problematiche di localizzazione e consapevolezza del contesto differenti da quelle proposte dallo scenario indoor per l'applicazione museale. In particolare, la localizzazione geografica può avvenire via GPS.

La guida in questione si pone come uno scenario a metà tra un sistema tradizionale e un sistema di e-learning. Essa, in particolare è pensata, come un "gioco" per guidare scolaresche alla scoperta del sito.

Il gioco è organizzato in tre fasi:

Fase iniziale. Il game master spiega il gioco e le sue regole. Quando gli studenti arrivano nel sito archeologico, vengono divisi in gruppi di 4/5 persone dal game master (che può essere, ad esempio, un insegnante). Ogni gruppo simula il comportamento di un nucleo tipico dell'epoca del sito (ad esempio, una *famiglia romana*) arrivata da poco sul sito stesso, avendo ricevuto alcuni appezzamenti e una casa nella colonia. Ogni famiglia ha un suo kit di gioco contenente:

- una mappa della città;
- il dispositivo mobile.

Fase di gioco. Ogni nucleo ha il compito di svolgere un certo numero di attività, presentate sotto forma di missioni, che consistono nell'esplorare il parco archeologico per riconoscere i luoghi che rappresentano la soluzione delle varie missioni. Per avere successo nel portare a termine le missioni, gli studenti devono formulare ipotesi, discuterle, tornare sui propri passi e correggerle.

Il testo di ogni missione appare sullo schermo del dispositivo mobile. È possibile affrontare le missioni nell'ordine in cui vengono proposte, ma al gruppo è lasciata la libertà di cambiarlo a seconda di altri criteri personali (posizione all'interno del sito, livello di difficoltà, ecc).

Quando il gruppo ritiene di aver individuato un luogo che rappresenta l'obiettivo di una missione, inserisce la risposta usando il dispositivo mobile: alla pressione di un tasto, il dispositivo invierà la posizione GPS del luogo in cui si trova il gruppo; questa verrà confrontata con la posizione del luogo che rappresenta la soluzione corretta per quella missione. Nel caso in cui il gruppo non riuscisse a trovare il luogo esatto, potrà chiedere al sistema un aiuto che sarà mostrato a video e conterrà piccoli suggerimenti per individuare l'edificio, insieme ad ulteriori informazioni storiche. Se il luogo indicato è l'obiettivo corretto della missione, sul dispositivo verrà riprodotta la ricostruzione 3D dello stesso; invece, nel caso in cui il gruppo non abbia eseguito correttamente la missione, sul dispositivo verrà visualizzato un messaggio di risposta non corretta, con l'invito ad approfondire i motivi dell'insuccesso nella fase di debriefing.

L'esperienza di gioco viene potenziata tramite l'utilizzo di suoni ambientali contestualizzati (il vociare delle persone, il rumore dei carri, ecc.) in grado sia di rendere le ricostruzioni più immersive sia di giocare in un contesto di realtà aumentata. Può essere anche sperimentato l'uso di questi suoni ambientali come "suggerimenti" contestuali per la risoluzione delle missioni.

L'interazione con il sistema può avvenire anche tramite l'uso di sketch da disegnare sul display del dispositivo, utilizzando lo stilo in dotazione: tramite semplici segni grafici sarà possibile impartire comandi per spostarsi attraverso le varie schermate dell'applicazione, chiedere un

aiuto, oppure per ruotare, zoomare o compiere altre manipolazioni sulle ricostruzioni 3D.

L'hardware necessario per l'esecuzione della fase di gioco è il seguente:

- auricolari per palmare o smartphone;
- palmare o smartphone dotato di antenna GPS interna o esterna;
- notebook dotato di scheda wireless, esso fungerà da server.

Fase di debriefing. Si riflette sul gioco e sull'esperienza appena vissuta. Al termine della fase di gioco, tutti i gruppi vengono riuniti: il game master controlla i risultati del gioco e si discute insieme sul ruolo interpretato dal gruppo, al fine di portare alla luce gli aspetti significativi della cultura che si è studiata.

Uso di suoni ambientali. Mentre i gruppi eseguono le missioni loro assegnate, i dati relativi alla loro posizione vengono inviati in real-time al server che ne elabora il contenuto. L'elaborazione consiste nel calcolare le distanze geografiche tra il client in dotazione a ciascun gruppo e i punti di interesse nel parco.

In questa maniera, ciascun gruppo potrà ascoltare i suoni ambientali in maniera suggestiva e rivivere le emozioni di un tempo, con la particolarità di poter ascoltare ad un volume maggiore i suoni che sono più vicini rispetto ad altri. In questa ottica si vuole realizzare un applicativo che permetta di implementare, nella maniera più efficace possibile, quanto descritto precedentemente.

L’ambiente sonoro sarà costituito da un numero di fonti adeguato all’ambiente visitato. Ogni fonte sarà posizionata in concomitanza di una coordinata fisica GPS.

La costruzione di un ambiente permette al visitatore di avere una percezione totale dell’ambiente che lo circonda. Così facendo, durante il cammino, i suoni si moduleranno in maniera tale da consentire all’utente la percezione della distanza da un certo luogo di interesse. Durante l’attraversamento di aree del parco in cui non sono presenti obiettivi significativi dal punto di vista del gioco, verrà riprodotto un suono di sottofondo in grado di riprodurre una *”atmosfera sonora”* significativa di quell’area della città.

3.5 ITR e CHAT, perché integrare?

In questo paragrafo analizzeremo le motivazioni che hanno portato alla successiva integrazione di ITem Recommender nel progetto CHAT. Per realizzare questo obiettivo si farà riferimento allo scenario museale visto al paragrafo 3.4.1.

Come già visto in precedenza, lo scenario prevede, tramite una dotazione di dispositivi *wireless*, la fruizione di particolari contenuti relativi alle opere. Tramite i vari punti di connessione nelle stanze della pinacoteca, si potrà sapere in ogni istante in che sala si trova l’utente, il quale potrà interagire col sistema attraverso comandi particolari per ottenere i dati da lui desiderati.

Tutti questi dati, però, hanno la particolarità di essere *statici* e *non personalizzati*: questo significa che, per qualsiasi tipologia di utente, ad una particolare azione verrebbero restituiti gli stessi risultati.

Come *migliorare* la qualità dei risultati forniti dai dispositivi? Per venire incontro a questa domanda è utile proprio l'utilizzo di ITem Recommender. Introduciamo, dunque, un ipotetico scenario per la creazione di contenuti personalizzati sotto il punto di vista dell'*interesse di un utente verso determinate opere*. Condizione imprescindibile per la realizzazione di questo scenario è la corrispondenza fra gli ID degli item memorizzati in Fedora con gli ID di quelli memorizzati in ITR.

Fase 1: iscrizione al portale del museo. In una fase preliminare, un utente che desidera visitare un museo potrebbe iscriversi ad un portale ad esso dedicato. All'atto dell'iscrizione, dopo aver inserito i classici dati anagrafici (età, sesso, ecc.) e personali (istruzione, interessi, ecc.), utili per una prima fase di personalizzazione, ITR inserisce un nuovo utente nel sistema (*UserJDBC*.addUser); dunque, si potrebbe decidere di *votare* delle opere, scelte in maniera random in tutto il dataset memorizzato in *Fedora*.

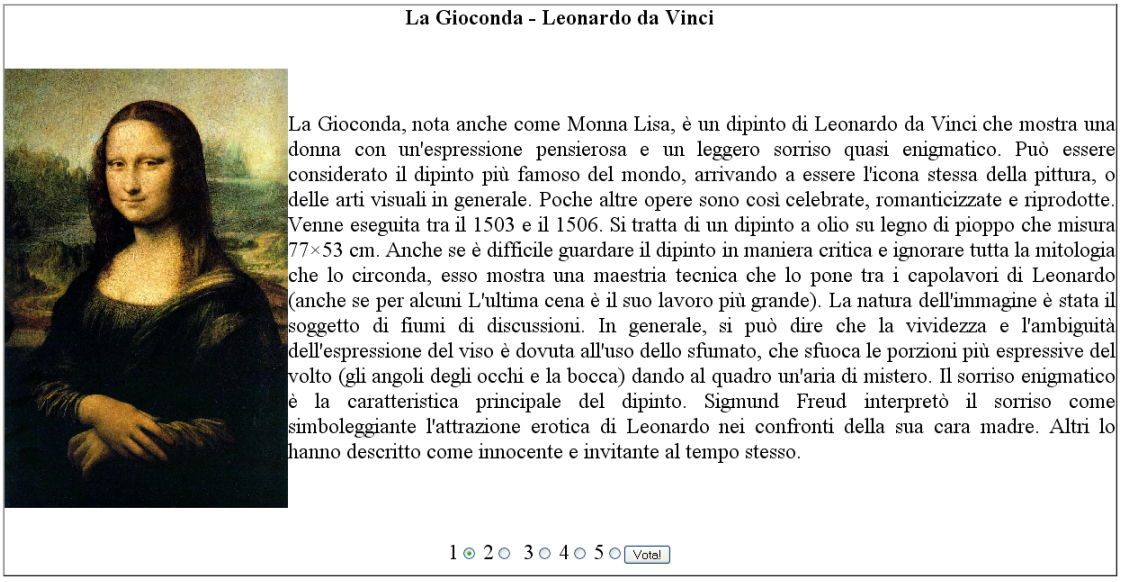

Figura 48 - Esempio di votazione di un'opera

Alla votazione di un'opera, ITR inserisce un nuovo voto nel sistema (*RatesJDBC.addRates*).

Fase 2: costruzione del profilo. Una volta terminata questa fase di *training*, ITR costruisce un profilo per l'utente calcolando, per gli *item* non votati, gli score relativi all'eventuale *gradimento* dell'item per l'utente.

```
<?xml version="1.0" encoding="UTF-8" ?>
<profile>
   <user>1</user>
   <category>classify</category>
   <probability like="0.4772727272727273" dislike="0.030303030303030304"
    size_like="28" size_dislike="3" />
   <vocabulary_like>
      <title>
    <feature value="lute" frequence="2" />
    <feature value="deposed" frequence="1" />
    <feature value="redeemer" frequence="2" />
    <feature value="platina" frequence="1" />
    <feature value="dominic" frequence="3" />
    <feature value="appoints" frequence="2" />
    <feature value="communion" frequence="1" />
    <feature value="erasmus" frequence="2" />
    <feature value="vincent" frequence="1" />
    <feature value="charity" frequence="1" />
    <feature value="benedict" frequence="1" />
    <feature value="constance" frequence="2" />
    <feature value="ercolanus" frequence="1" />
```

Figura 49 - Stralcio di un profilo in formato XML

Oltre al profilo, sempre utile per interazioni future, ITR salverà tutti i punteggi in una tabella, per recuperarli in seguito.

idUser	idItem	plike	pdislike	voted
4	34	0.9238535196...	0.0664703569...	0
4	35	0.9405836095...	0.0588479228...	0
4	32	0.8610964250...	0.0976601493...	0
4	33	0.8465055467...	0.0815538018...	0
4	38	0.9235144669...	0.0622810985...	0
4	39	0.9213398520...	0.0652579105...	0
4	36	0.9039488553...	0.0728706720...	0
4	37	0.8510663045...	0.0789926819...	0
4	42	0.8777669126...	0.0920211369...	0
4	43	0.9064299710...	0.0627920573...	0
4	40	0.8277248805...	0.0817525623...	0
4	41	0.8526786593...	0.1132309944...	0
4	44	0.8926589994...	0.0630948705...	0
4	45	0.8573968509...	0.0984638640...	0
4	1	0.2	0.8	1
4	2	0.2	0.8	1
4	3	0.6	0.4	1
4	4	0.8	0.2	1
4	5	0.6	0.4	1
4	6	0.4	0.6	1
4	7	0.8	0.2	1
4	8	0.8	0.2	1
4	9	0.6	0.4	1

Figura 50 - La tabella Classify del database di ITR

Fase 3: personalizzazione dei risultati. A questo punto, quando l'utente si recherà *fisicamente* presso il museo, oltre ad ottenere le classiche informazioni sulle opere, potrebbe ottenere informazioni aggiuntive ma molto importanti, quali, per esempio, le opere più interessanti o le stanze di maggiore gradimento da visitare.

4. Progetto e sviluppo di servizi per l'accesso alla Pinacoteca dei Musei Vaticani

In questo capitolo verrà tracciato lo sviluppo dei servizi per l'accesso ad un sito museale, come disposto dai capitolati relativi al progetto CHAT.

Dunque, si partirà dalla scelta del museo utilizzato per la progettazione, per passare poi ai servizi di supporto utilizzati per il trasferimento dei contenuti in Fedora, fino ad arrivare alla progettazione dei servizi, adattivi e non, per l'accesso multimodale alle opere.

4.1 La pinacoteca dei Musei Vaticani

La scelta del museo sul quale realizzare i serrvizi è ricaduta sulla Pinacoteca dei Musei Vaticani. [31] Le caratteristiche principali di questa struttura sono la suddivisione in sale e la sola presenza di quadri.

Figura 51 - Home Page della Pinacoteca dei Musei Vaticani

Nei prossimi paragrafi, prima dell'esposizione dei servizi vera e propria, verrà illustrata l'estrazione delle informazioni dal sito della Pinacoteca ed il successivo trasferimento degli stessi nei Digital Objects di Fedora.

4.2 Estrazione delle informazioni: PinacotecaRest

PinacotecaRest [33] è un servizio web REST, realizzato in Java con l'ausilio del framework Restlet [32]. Esso, utilizzando come sorgente dati alcuni robot sviluppati con OpenKapow, [34] un software di screen-scraping, si occupa di manipolare le informazioni acquisite dal sito della Pinacoteca e di restituirle in output in formato XML. Il servizio, inoltre, espone anche delle funzionalità di disambiguazione delle informazioni estratte attraverso un interfacciamento con il Web Service META (Multilanguage TExt Analyzer), sviluppato dal Dipartimento di Informatica dell'Università di Bari.

L'obiettivo generale del servizio, in modo assolutamente semplicistico, è di estrarre tutte le informazioni contenute nel sito ufficiale della Pinacoteca e esporle sul Web in formato XML (opportunamente disambiguate mediante l'interfacciamento con il tool META) attraverso un servizio Web REST.

Un particolare non banale è rappresentato dall'assenza di API. Il sito della Pinacoteca, allo stato attuale, non espone dei metodi standard per accedere al sistema ed acquisire le informazioni. Questa mancanza ha portato, nella fase di progettazione del sistema, alla necessità di aggiungere un ulteriore step, integrando nella componente anche dei meccanismi di screen-scraping (estrazione di informazioni da una pagina web) che saranno descritti dettagliatamente in seguito.

Il servizio PinacotecaRest espone dunque le seguenti funzionalità:

- Estrazione delle informazioni su tutte le opere presenti all'interno del sito;
- Disambiguazione delle descrizioni associate a ciascuna opera;
- Estrazione della BOC (Bag of Concept) associata alla descrizione attraverso l'interfacciamento con il tool di disambiguazione Meta.

Seguendo le linee generali dei servizi Web REST , l'accesso a tutte queste informazioni non avviene attraverso i classici scambi di messaggi SOAP, ma attraverso semplici richieste HTTP (di tipo "GET", essenzialmente) inoltrate verso specifici URL strutturati secondo la logica REST. Nello specifico:

- *http://host:porta/pinacotecaRest*

  URL di Base del Servizio

- *http://host:porta/pinacotecaRest/opus*

  Tutte le opere in formato XML

- *http://host:porta/pinacotecaRest/opus/{n}*

  Informazioni XML sulla n-esima opera

- *http://host:porta/pinacotecaRest/opus/{n}/meta*

  Descrizione disambiguata

- *http://host:porta/pinacotecaRest/opus/{n}/boc*

  Bag Of Concept associata all'opera *n*

4.3 Trasferimento delle informazioni: Pinacoteca2Fedora

Come precedentemente accennato, una volta terminata la fase di estrazione dei dati dal sito della Pinacoteca Vaticana, risulta necessario trasformali in un opportuno formato e successivamente effettuare l'ingest in Fedora. Queste funzioni sono delegate ad un client scritto in linguaggio Java. [35]

Elenchiamo schematicamente le funzioni che esso svolge:

- Connessione al server di Fedora;
- Invocazione del Web Service per l'estrazione delle informazioni relative alle opere presenti nella Pinacoteca Vaticana;
- Creazione in Fedora dei Digital Object relativi alle stanze, agli autori ed alle opere;
- Aggiunta dei Datastream relativi alla descrizione testuale dell'opera.

Allo stato attuale, i dati estratti sono quelli della versione inglese del sito della Pinacoteca Vaticana.

Il client è completamente parametrizzato. Tali parametri vengono estratti da alcuni file di configurazione in formato XML. Il client, inoltre, per le trasformazioni nei formati idonei all'ingest in Fedora, si avvale di fogli di stile XSL.

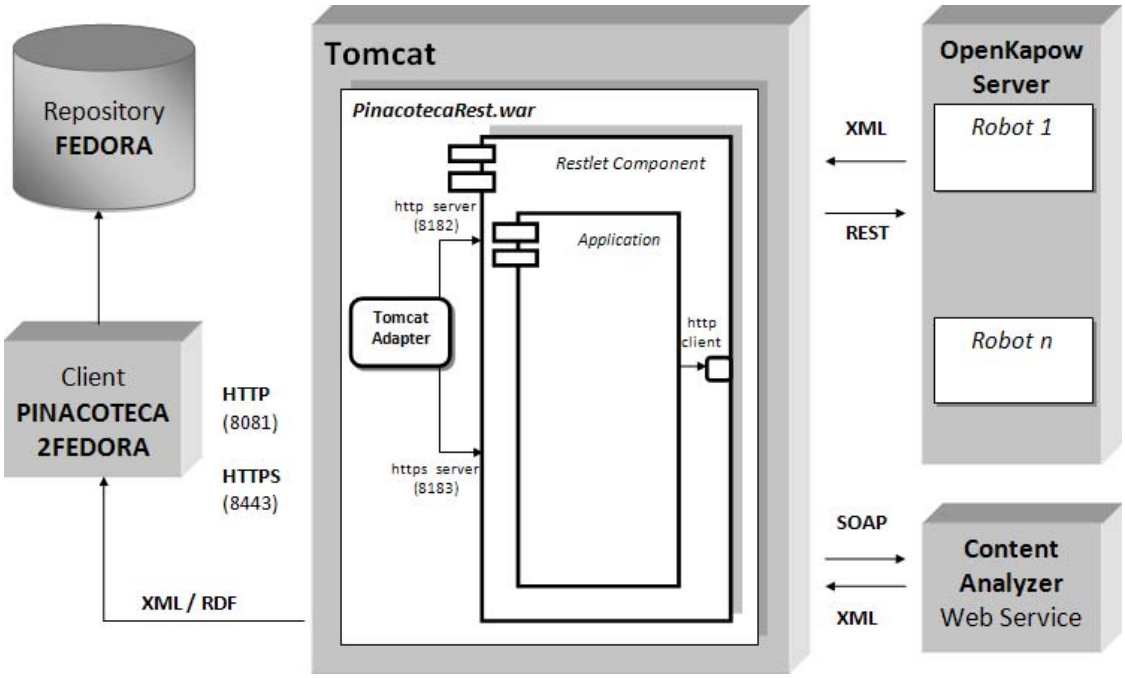

Figura 52 - Architettura generale di PinacotecaRest e Pinacoteca2Fedora

4.4 Task per l'applicazione in ambito museale

Per un'applicazione relativa allo scenario beni culturali che simuli la visita di un museo, sono stati previsti una serie di task (ovvero comandi multimodali). È importante sottolineare il fatto che la natura del task richiesto dall'utente è identificata in ambito "*fusion*" a monte del processo di Machine Learning.

I task indicati sono relativi esclusivamente all'ambito museale. Eventuali dimostratori relativi ad altri contesti (ad es. e-learning) dovranno fare uso di altri task più adatti al relativo contesto.

Si immagini dunque che ad un utente – come detto nei precedenti capitoli – all'ingresso del museo venga fornito un dispositivo mobile in grado di gestire diversi canali di input (touch screen ed acquisizione audio). L'utente inizia la propria visita spostandosi attraverso le diverse stanze del museo ed essendo sempre localizzato da sensori di posizione. Inoltre, tramite collegamento wireless (nel caso specifico Bluetooth)

l’applicazione è in grado di reperire informazioni di carattere ambientale, quali luminosità e rumorosità del luogo.

All’ingresso di ogni sala i contenuti presentati all’utente si adattano spontaneamente mostrando il catalogo delle opere presenti all’interno dell’area. L’utente può selezionare con lo stilo l’opera a cui è interessato e chiedere ulteriori informazioni tramite relativo comando vocale.

Si tratta di uno scenario semplice che può essere complicato con l’aggiunta di comandi e contenuti. In questa fase, i task individuati per i contenuti testuali sono riportati nella tabella che segue.

Nome task	**Descrizione**	**Parametri**
Location	È il comando con il quale si richiede di ricevere contenuti relativi ad una stanza specifica del museo.	1. Il numero identificativo della stanza (PID dell’oggetto stanza). 2. (eventuale) la modalità con cui l’utente specifica di voler ricevere i contenuti richiesti.
Info	È il comando con il quale si richiede di ricevere informazioni relativamente ad un’opera o ad un autore del museo.	1. L’identificativo (PID) dell’oggetto selezionato. 2. (eventuale) la modalità con cui l’utente specifica di voler ricevere i contenuti richiesti.

Tabella 47 - Servizi di CHAT (1/2)

Nome task	Descrizione	Parametri
More	È il comando con il quale si richiede di ricevere ulteriori informazioni (livello di approfondimento superiore a quello del comando **info**). L'oggetto dell'approfondimento può essere tipicamente un'opera del museo.	1. L'identificativo (PID) dell'oggetto selezionato. 2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.
Other	È il comando con il quale si richiede di visualizzare altre opere, associate in qualche modo a quella indicata.	1. L'identificativo (PID) dell'oggetto selezionato (un'opera o un autore in questo caso). 2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.
Author	È il comando con il quale si richiede di ricevere informazioni sull'autore di un'opera o sugli autori presenti in una sala.	1. L'identificativo (PID) dell'oggetto selezionato 2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

Tabella 48 - Servizi di CHAT (2/2)

Nei paragrafi a seguire verrà mostrata la realizzazione di tali servizi, e il modo in cui i risultati attesi cambino in funzione della fruizione di ITR all'interno degli stessi. Ai servizi già decisi in fase preliminare, è stato aggiunto il servizio *classify*, il quale serve a mostrare le classificazioni degli item non votati per ogni utente.

4.5 I servizi realizzati

In questo paragrafo verranno mostrati i servizi realizzati. Per fare ciò, è stato deciso di utilizzare la tecnologia REST [32] per implementare un Web Services di facile accesso, ITRRest. Difatti, basteranno delle semplici chiamate http con dei parametri passati nella barra degli indirizzi stessa (conoscendo la sintassi dei servizi stessi) per ottenere, in formato XML, le informazioni desiderate.

4.5.1 Location

Il servizio *location* è il comando con il quale si richiede di ricevere contenuti relativi ad una stanza specifica del museo. I parametri in input sono i seguenti:

1. Il numero identificativo della stanza (PID dell'oggetto stanza).
2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

Location

/location	Tutte le location
/location/room:{n}	Opere nella location *n*
/location/user:{n}	Tutte le location adattate all'utente *n*
/location/user:{n}/ordered	Tutte le location ordinate per l'utente *n*
/location/user:{n}/ordered:{m}	Prime *m* location ordinate per l'utente *n*
/location/user:{n}/room:{m}	Location *m* per l'utente *n*

Figura 53 - Sintassi per il servizio location

Nella figura sopra, è presente la sintassi per accedere al servizio location. Vediamo nel dettaglio ciascuno dei servizi.

Sintassi: http://host:porta/itrRest/location

Restituisce tutte le opere presenti in tutte le location della Pinacoteca.

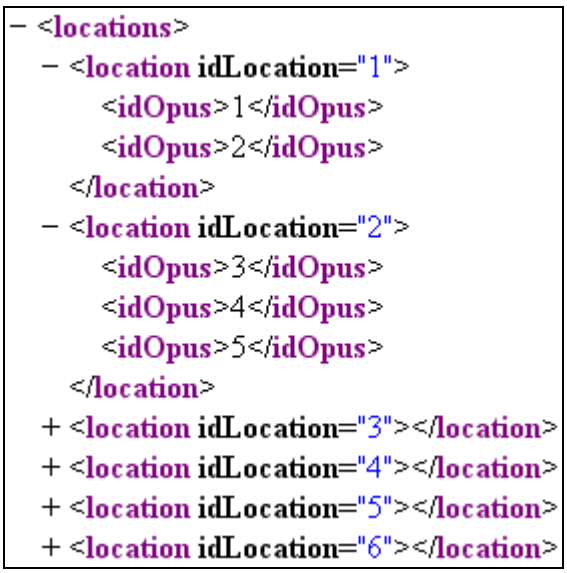

Figura 54 - Output per il servizio

Sintassi: http://host:porta/itrRest/location/room:{n}

Restituisce tutte le opere presenti nella location con ID *n*.

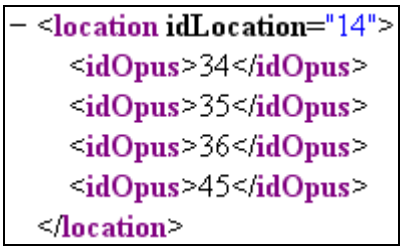

Figura 55 - Output per il servizio

Sintassi: http://host:porta/itrRest/location/user:{n}

Restituisce tutte le opere presenti in tutte le location *adattate* all'utente con ID *n*. Vale a dire, alle opere sarà associato uno score *like* e *dislike* e le stesse saranno *rankate* in ordine decrescente di interesse per quell'utente.

Inoltre, per ogni opera verrà indicato se l'utente ha espresso un voto in fase di *training* o meno.

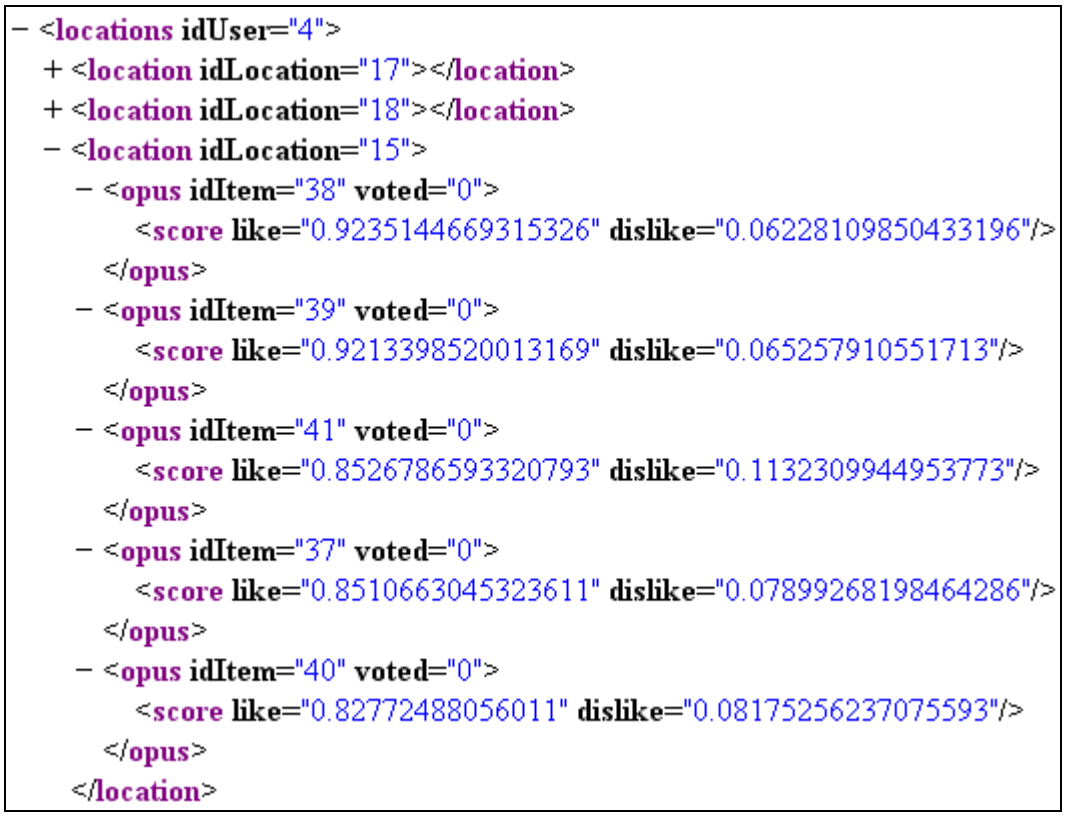

Figura 56 - Output per il servizio

Sintassi: http://host:porta/itrRest/location/user:{n}/ordered

Restituisce tutte le opere presenti in tutte le location *adattate* all'utente con ID *n ordinate in base al voto medio like delle opere presenti per ciascuna stanza*. Vale a dire, alle opere sarà associato uno score *like* e *dislike* e le stesse saranno *rankate* in ordine decrescente di interesse per quell'utente; le stanze, inoltre, verranno mostrate per ordine di importanza per quell'utente. Infine, per ogni opera verrà indicato se l'utente ha espresso un voto in fase di *training* o meno.

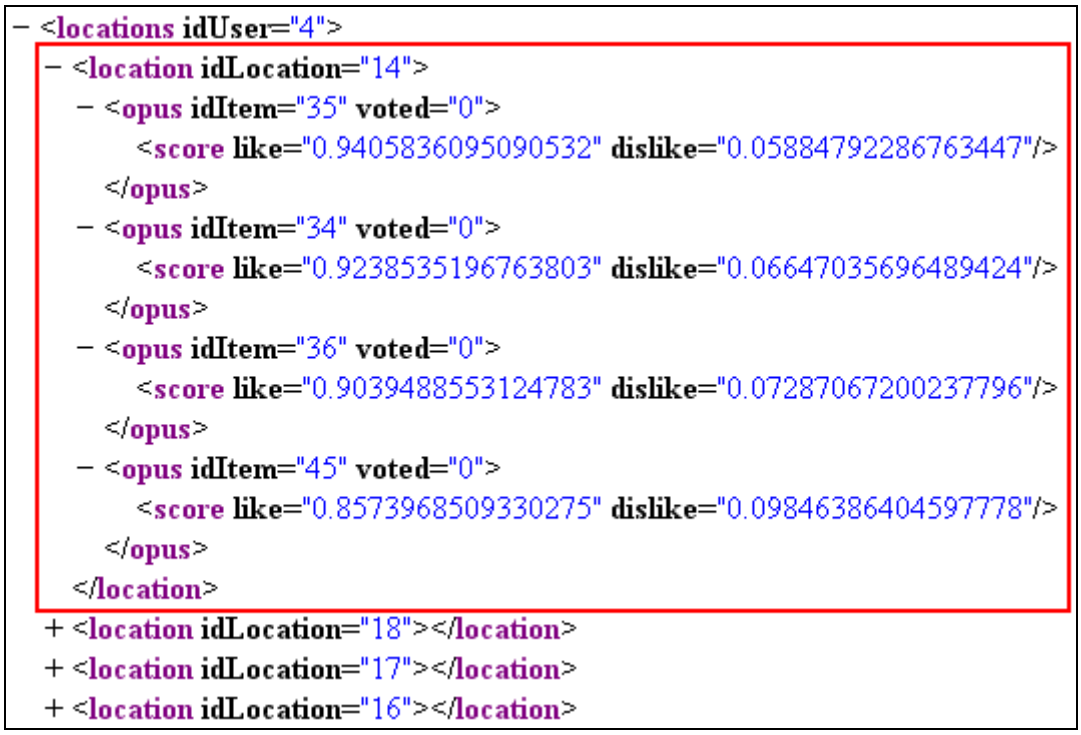

Figura 57 - Output del servizio; in rosso l'opera più importante

Sintassi: http://host:porta/itrRest/location/user:{n}/ordered:{m}

Restituisce tutte le opere presenti nelle prime *m* location *adattate* all'utente con ID *n ordinate in base al voto medio like delle opere presenti per ciascuna stanza*. Vale a dire, alle opere sarà associato uno score *like* e *dislike* e le stesse saranno *rankate* in ordine decrescente di interesse per quell'utente; le stanze, inoltre, verranno mostrate per ordine di importanza per quell'utente. Infine, per ogni opera verrà indicato se l'utente ha espresso un voto in fase di *training* o meno.

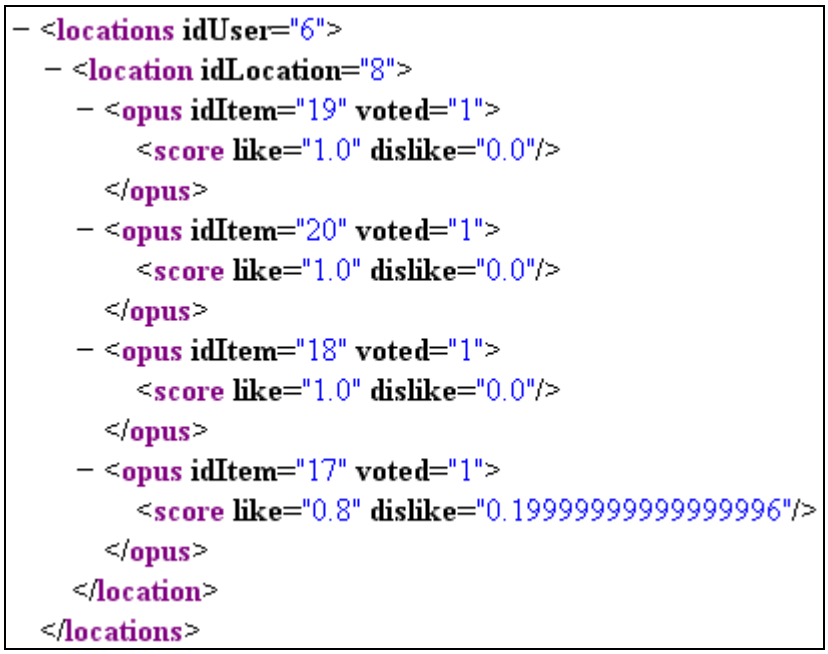

Figura 58 - Output del servizio

Sintassi: http://host:porta/itrRest/location/user:{n}/room:{m}

Restituisce tutte le opere presenti nella location con ID *m* *adattate* all'utente con ID *n*. Vale a dire, alle opere sarà associato uno score *like* e *dislike* e le stesse saranno *rankate* in ordine decrescente di interesse per quell'utente. Inoltre, per ogni opera verrà indicato se l'utente ha espresso un voto in fase di *training* o meno.

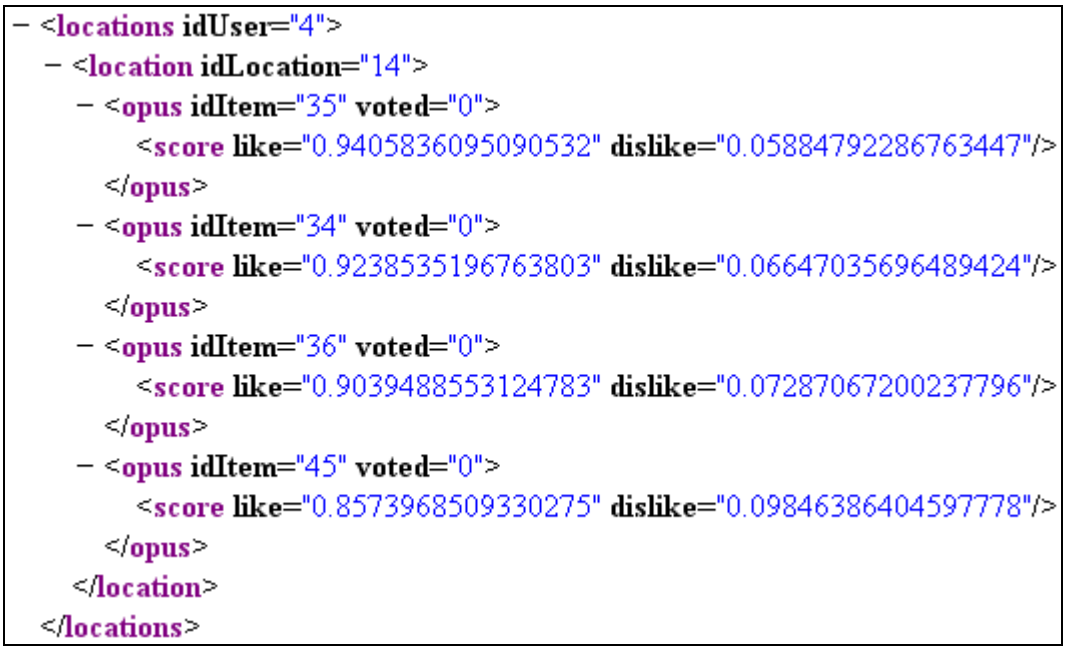

Figura 59 - Output del servizio

4.5.2 Info

Il servizio *info* è il comando con il quale si richiede di ricevere informazioni relativamente ad un'opera o ad un autore del museo. I parametri di input sono i seguenti:

1. L'identificativo (PID) dell'oggetto selezionato.
2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

Info

/info/author:{n}	Info sull'autore *n*
/info/opus:{n}	Info sull'opera *n*

Figura 60 - Sintassi per il servizio info

Nella figura precedente, è presente la sintassi per accedere al servizio info. Vediamo nel dettaglio ciascuno dei servizi.

Sintassi: http://host:porta/itrRest/info/author:{n}

Restituisce informazioni sull'autore con ID *n*.

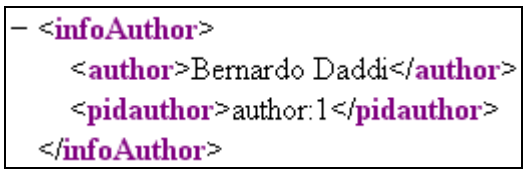

Figura 61 - Output del servizio

Sintassi: http://host:porta/itrRest/info/opus:{n}

Restituisce informazioni sull'opera con ID *n*.

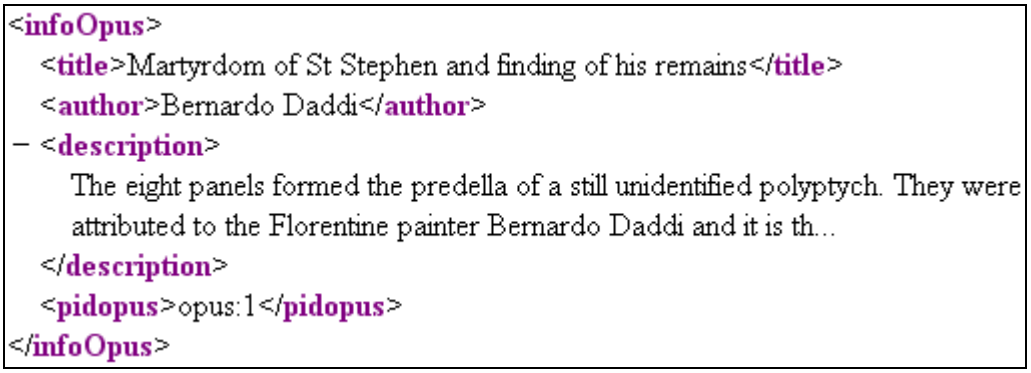

Figura 62 - Output del servizio

4.5.3 More

Il servizio *more* è il comando con il quale si richiede di ricevere ulteriori informazioni (livello di approfondimento superiore a quello del comando info). L'oggetto dell'approfondimento può essere tipicamente un'opera del museo. I parametri in input sono i seguenti:

1. L'identificativo (PID) dell'oggetto selezionato.
2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

More

/more/opus:{n}	Info più dettagliate sull'opera *n*

Figura 63 - Sintassi per il servizio more

Nella figura sopra, è presente la sintassi per accedere al servizio info. Vediamo nel dettaglio il servizio.

Sintassi: http://host:porta/itrRest/more/opus:{n}

Restituisce informazioni approfondite sull'opera con ID *n*.


```
− <moreOpus>
    <title>Martyrdom of St Stephen and finding of his remains</title>
    <author>Bernardo Daddi</author>
  − <description>
      The eight panels formed the predella of a still unidentified polyptych. They were
      attributed to the Florentine painter Bernardo Daddi and it is thought that they were
      painted around 1345, as part of the later activity of the artist. The small paintings
      illustrate the martyrdom of St Stephen and the story of the finding of his remains
      according to the medieval tale of the Legenda Aurea by Jacopo da Varagine. They
      begin with the Stoning of the Saint, followed by: The apparition in a dream of St
      Gamaliele, St Paul's master, to St Lucian, revealing where his body is buried together
      with those of Abibus, his son, Nicodemus and St Stephen; St Lucian tells John of his
      vision, and John, Patriarch of Jerusalem, has the bodies sought for; the Finding of the
      bodies of Sts Lucian, Abibus, Nicodemus and Stephen in the place indicated by
      Gamaliele; the Transfer of the bodies of the Saints to Jerusalem, where that of St
      Stephen miraculously healed Eudossias, daughter of the emperor Theodosius,
      possessed by an evil spirit; the Second transfer to Rome; the Reunion of the body of
      St Stephen with that of St Laurence in Rome and, lastly, the Needy who implore
      miracles on the tomb of St Laurence and St Stephen.
    </description>
    <pidopus>opus:1</pidopus>
  </moreOpus>
```


Figura 64 - Output del servizio

4.5.4 Other

Il servizio *other* è il comando con il quale si richiede di visualizzare altre opere, associate in qualche modo a quella indicata. I parametri in input sono i seguenti:

1. L'identificativo (PID) dell'oggetto selezionato (un'opera o un autore in questo caso).
2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

Other

/other/opus:{n}	Altre opere presenti nella stanza in cui è presente l'opera *n*
/other/author:{n}	Altre opere riguardanti l'autore *n*

Figura 65 - Sintassi per il servizio other

Nella figura sopra, è presente la sintassi per accedere al servizio other. Vediamo nel dettaglio ciascuno dei servizi.

Sintassi: http://host:porta/itrRest/other/opus:{n}

Restituisce le altre opere presenti nella stessa stanza in cui è presente l'opera con ID *n*.

```
− <otherOpus idOpus="1" idLocation="1">
  − <opus idOpus="2">
      <title>Giudizio Finale</title>
      <author>Nicolò e Giovanni</author>
    </opus>
  </otherOpus>
```

Figura 66 - Output del servizio

Sintassi: http://host:porta/itrRest/other/author:{n}

Restituisce le altre opere presenti nella pinacoteca il cui autore è quello con ID *n*.

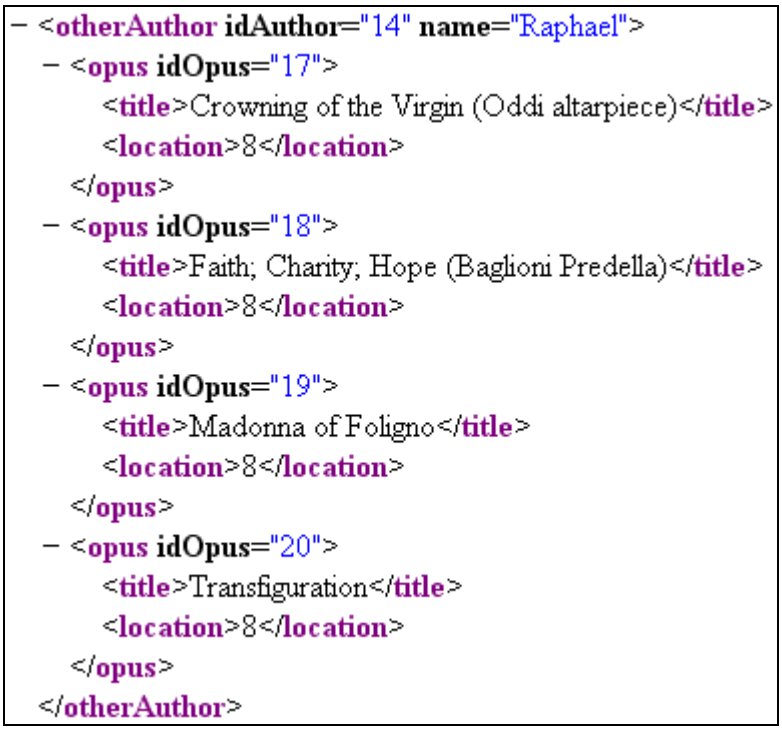

Figura 67 - Output del servizio

4.5.5 Author

Il servizio *author* è il comando con il quale si richiede di ricevere informazioni sull'autore di un'opera o sugli autori presenti in una sala. I parametri in input sono i seguenti:

1. L'identificativo (PID) dell'oggetto selezionato.
2. (eventuale) la modalità con cui l'utente specifica di voler ricevere i contenuti richiesti.

Author

/author/location:{n}	Autori presenti nella location *n*
/author/opus:{n}	Autore dell'opera *n*

Figura 68 - Sintassi per il servizio author

Nella figura sopra, è presente la sintassi per accedere al servizio author. Vediamo nel dettaglio ciascuno dei servizi.

Sintassi: http://host:porta/itrRest/author/location:{n}

Restituisce l'elenco degli autori presenti nella location con ID *n*.

```
- <authorLocation idLocation="14">
  - <opus idOpus="34">
      <author>Baciccia</author>
      <pidauthor>author:28</pidauthor>
    </opus>
  - <opus idOpus="35">
      <author>Sassoferrato</author>
      <pidauthor>author:29</pidauthor>
    </opus>
  - <opus idOpus="36">
      <author>Daniel Seghers and Erasmus II Quellin</author>
      <pidauthor>author:30</pidauthor>
    </opus>
  - <opus idOpus="45">
      <author>Carlo Maratta</author>
      <pidauthor>author:39</pidauthor>
    </opus>
  </authorLocation>
```

Figura 69 - Output del servizio

Sintassi: http://host:porta/itrRest/author/opus:{n}

Restituisce l'autore dell'opera con ID *n*.

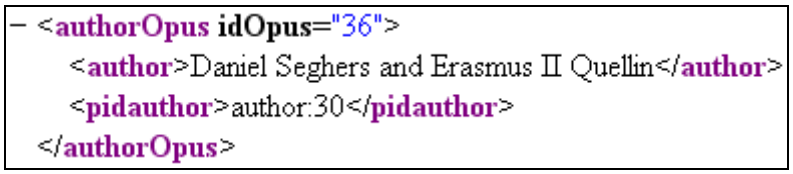

Figura 70 - Output del servizio

4.5.6 Classify

Il servizio *classify* è stato aggiunto a quelli predefiniti nel progetto, per fare in modo di usufruire delle classificazioni effettuate con ITR e restituire, in ordine di interesse, le opere per determinati utenti.

Classify

/classify	Classificazione per tutti gli utenti
/classify/user:{n}	Classificazione per l'utente *n*
/classify/user:{n}/num:{m}	Classificazione delle prime *m* opere dell'utente *n*

Figura 71 - Sintassi per il servizio classify

Nella figura sopra, è presente la sintassi per accedere al servizio classify. Vediamo nel dettaglio ciascuno dei servizi.

Sintassi: http://host:porta/itrRest/classify

Restituisce tutte le classificazioni già memorizzate nel database di ITR.

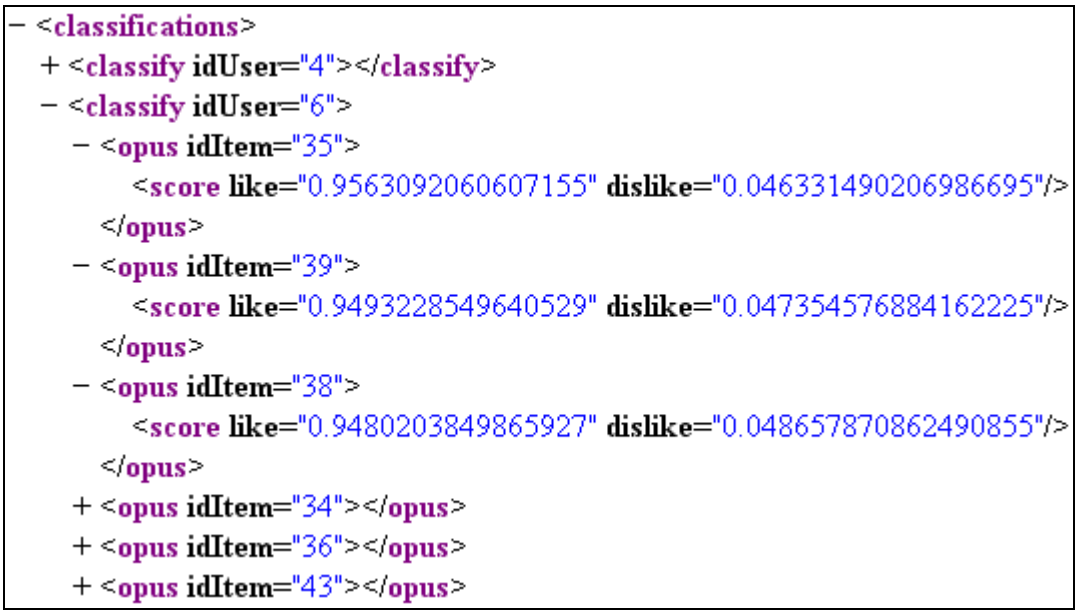

Figura 72 - Output del servizio

Sintassi: http://host:porta/itrRest/classify/user:{n}

Restituisce tutte le classificazioni già memorizzate nel database di ITR per l'utente *n*, vale a dire, verranno restituite le opere classificate per l'utente *n* e *rankate* in ordine decrescente di interesse.

Figura 73 - Output del servizio

Sintassi: http://host:porta/itrRest/classify/user:{n}/num:{m}

Restituisce le prime *m* classificazioni già memorizzate nel database di ITR per l'utente *n*, vale a dire, verranno restituite le *m* opere classificate per l'utente *n*, *rankate* in ordine decrescente di interesse.

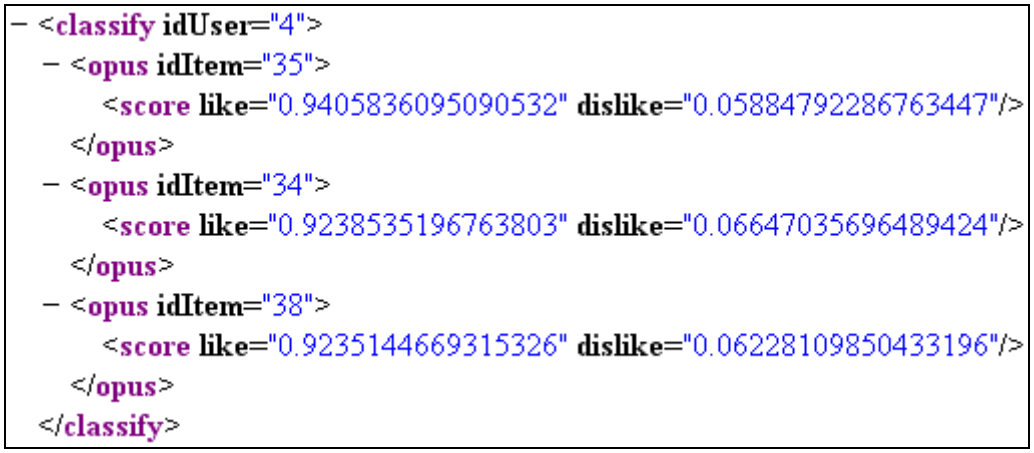

Figura 74 - Output del servizio

4.6 Diagramma dei package

Figura 75 - Diagramma dei package di ITRRest

Tramite la figura appena mostrata, andiamo a visualizzare la struttura del Web Service REST.

A parte il package *main*, che contiene la pagina principale del Web Service permettendo di accedere a tutti i sottosistemi, è stato realizzato un package diverso per ogni servizio di CHAT realizzato. Importante, infine, il package *util*, che presenta classi per manipolare *database*, *Digital Objects di Fedora*, *file di configurazione*, *stringhe*.

4.6.1 Package util

Il package util, come già detto, serve a manipolare tuple del database di ITR, Digital Objects di Fedora, file di configurazione e stringhe in genere.

ClassifyRecord è una struttura d'appoggio, utile a memorizzare le classificazioni ordinabili tramite il campo *comparation* in essa presente

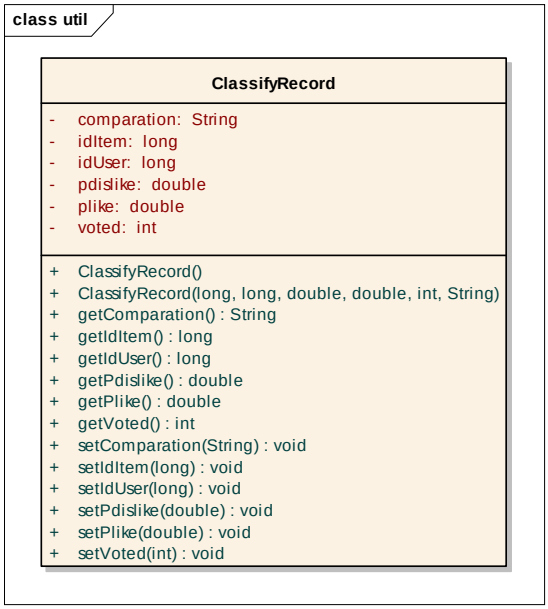

Figura 76 - ClassifyRecord

Metodi

Nome	Descrizione
ClassifyRecord()	Costruttore
ClassifyRecord(long, long, double, double, int, String)	Costruttore che seta tutti I campi della classe coi valori forniti in input
getComparation(): String	Ritorna il campo comparation
getIdItem(): long	Ritorna l'ID dell'item
getIdUser(): long	Ritorna l'ID dell'utente
getPdislike(): double	Ritorna lo score dislike
getPlike(): double	Ritorna lo score like
getVoted(): int	Ritorna il flag voted
setComparation (String): void	Setta il campo comparation col valore fornito in input
setIdItem(long): void	Setta l'ID dell'item col valore fornito in input
setIdUser(long): void	Setta l'ID dell'utente col valore fornito in input
getPdislike(double): void	Setta lo score dislike col valore fornito in input
getPlike(double): void	Setta lo score like col valore fornito in input
getVoted(int): void	Sett ail flag voted col valore fornito in input

Tabella 49 - Metodi della classe ClassifyRecord

Configuration è la classe che si occupa di manipolare il file di configurazione in ITRRest

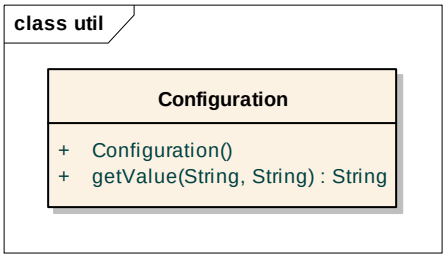

Figura 77 - Configuration

Metodi

Nome	Descrizione
Configuration()	Costruttore
getValue(String, String)	Ritorna il valore del nodo dato in input dal file di configurazione

Tabella 50 - Metodi della classe Configuration

DbFunctions si occupa di raggruppare tutte le funzioni di gestione e manipolazione di tuple del database di ITR (1/3)

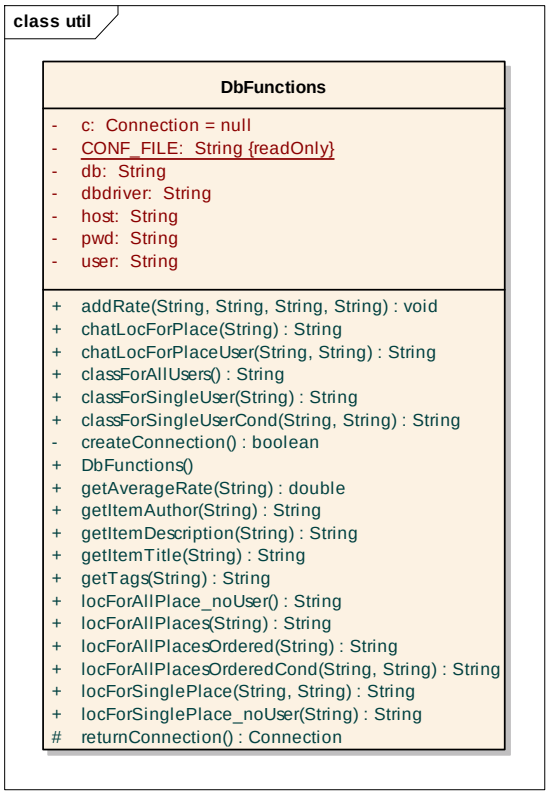

Figura 78 - DbFunctions

Metodi (1/3)

Nome	Descrizione
DbFunctions()	Costruttore
addRate(String, String, String, String): void	Aggiunge un nuovo voto al database
chatLocForPlace (String): String	Restituisce un file XML che rappresenta le opere per la location data in input
chatLocForPlaceUser (String, String): String	Restituisce un file XML che rappresenta le oepre per la location e l'utente dati in input
classForAllUsers(): String	Restituisce un file XML che rappresenta le classificazioni per tutti gli utenti
classForSingleUser (String): String	Restituisce un file XML che rappresenta la classificazione per l'utente dato in input
classForSingleUser Cond(String, String): String	Restituisce un file XML che rappresenta la classificazione per le prime *m* opere per l'utente dato in input

Tabella 51 - Metodi della classe DbFunctions (1/3)

DbFunctions si occupa di raggruppare tutte le funzioni di gestione e manipolazione di tuple del database di ITR (2/3)

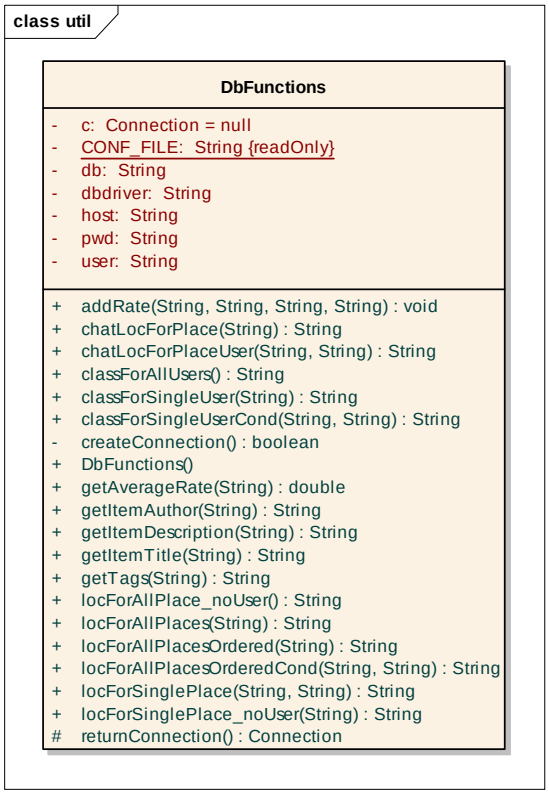

Figura 79 - DbFunctions

Metodi (2/3)

Nome	Descrizione
create Connection(): boolean	Crea la connessione al database
getAverageRate (String): double	Ritorna il voto medio per l'utente con ID dato in input
getItemAuthor (String): String	Ritorna l'autore per l'item con ID dato in input
getItemDescription (String): String	Ritorna la descrizione per l'item con ID dato in input
getItemTitle(String): String	Ritorna il titolo per l'item con ID dato in input
getTags(String): String	Ritorna i tags per l'item con ID dato in input
locForAllPlaces (String): String	Restituisce un file XML che contiene le opere di tutte le location per l'utente con ID dato in input
locForAllPlaces Ordered(String): String	Restituisce un file XML che contiene le opere di tutte le location ordinate per voto medio per l'utente con ID dato in input
locForAllPlaces OrderedCond(String, String): String	Restituisce un file XML che contiene le opere delle prime *m* location ordinate per voto medio per l'utente con ID dato in input

Tabella 52 - Metodi della classe DbFunctions (2/3)

DbFunctions si occupa di raggruppare tutte le funzioni di gestione e manipolazione di tuple del database di ITR (3/3)

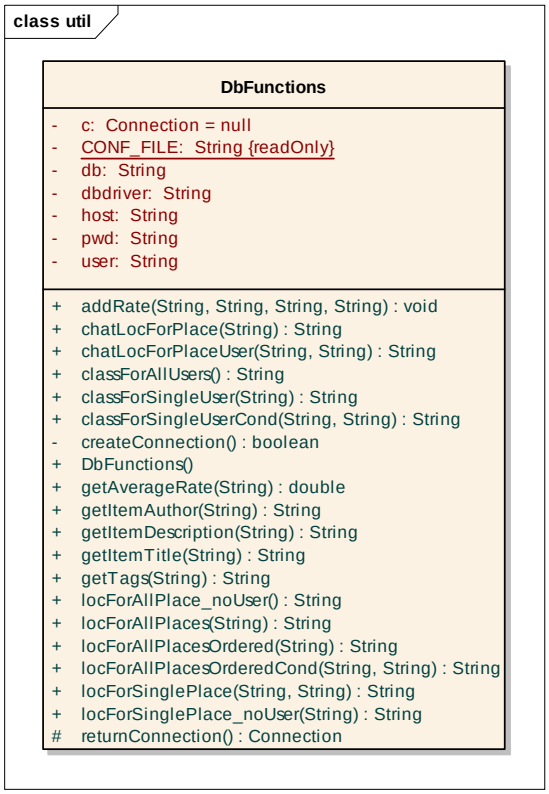

Figura 80 - DbFunctions

Metodi (3/3)

Nome	Descrizione
locForSinglePlace (String): String	Restituisce un file XML che contiene le opera della location con ID dato in input
locForAllPlace_ noUser(): String	Restituisce un file XML che contiene le opere presenti in tutte le location
locForSinglePlace_ noUser(String): String	Restituisce un file XML che contiene le opere presenti nella location con ID dato in input
returnConnection(): Connection	Ritorna la connessione al database

Tabella 53 - Metodi della classe DbFunctions (3/3)

FedoraFunctions è la classe che si occupa di estrarre contenuti dai Digital Objects di Fedora (1/2)

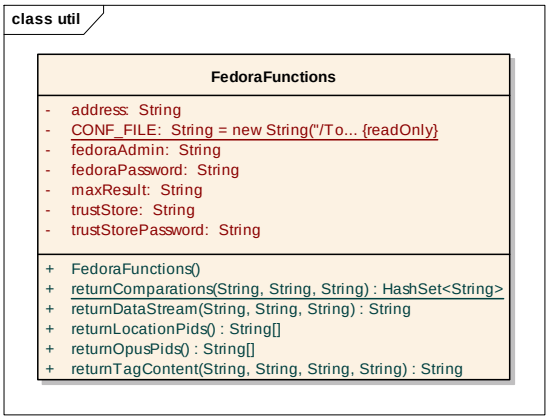

Figura 81 - FedoraFunctions

Metodi (1/2)

Nome	Descrizione
FedoraFunctions()	Costruttore
returnComparations (String, String, String): HashSet <String>	Ritorna un insieme di stringhe rappresentanti tutti i campi di comparazione estratti dai datastreams di Fedora
returnDataStream (String, String, String): String	Ritorna il datastream con id entità, pid e nome dati in input

Tabella 54 - Metodi della classe FedoraFunctions (1/2)

FedoraFunctions è la classe che si occupa di estrarre contenuti dai Digital Objects di Fedora (2/2)

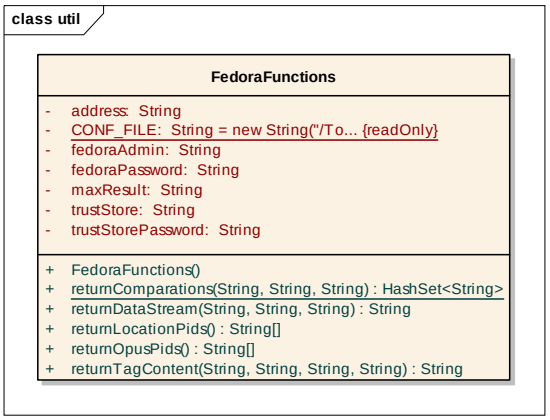

Figura 82 - FedoraFunctions

Metodi (2/2)

Nome	Descrizione
returnLocationPids(): String[]	Ritorna un array di PID delle location estratti dai DO di Fedora
returnOpusPids(): String[]	Ritorna un array di PID delle opere estratti dai DO di Fedora
returnTagContent (String, String, String, String): String	Ritorna il contenuto di un tag di un file XML presente dentro un datastream di un DO di Fedora

Tabella 55 - Metodi della classe FedoraFunctions (2/2)

LocationItem è la struttura di appoggio a LocationStruct e realizza un record con id location, numero di item per quella location, media dello score di like (1/2)

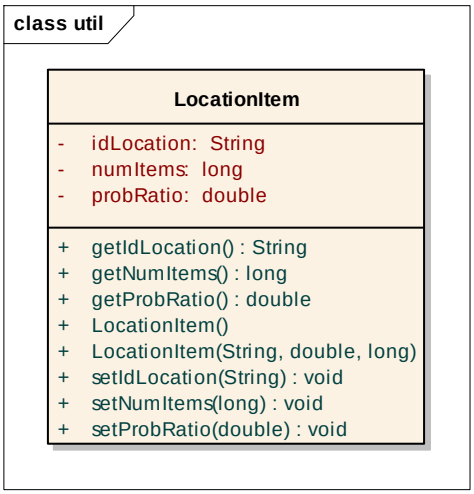

Figura 83 - LocationItem

Metodi (1/2)

Nome	Descrizione
LocationItem()	Costruttore
LocationItem(String, double, long)	Costruttore che inizializza i campi coi valori dati in input
getIdLocation(): String	Ritorna l'ID della location
getNumItems(): long	Ritorna il numero di items per la location
getProbRatio(): double	Ritorna la media dello score like per gli item
setIdLocation(String): void	Setta l'ID della location col valore dato in input

Tabella 56 - Metodi della classe LocationItem (1/2)

LocationItem è la struttura di appoggio a LocationStruct e realizza un record con id location, numero di item per quella location, media dello score di like (2/2)

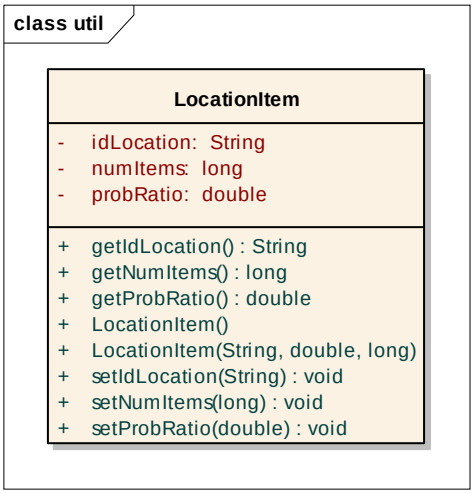

Figura 84 - LocationItem

Metodi (2/2)

Nome	Descrizione
setNumItems(long): void	Setta il numero di itelm col valore dato in input
getProbRatio(double): void	Setta la media dello score di like col valore dato in input

Tabella 57 - Metodi della classe LocationItem (2/2)

LocationStruct è la struttura che realizza i servizi di location. Tramite un campo di appoggio che ordina le location per importanza, costruisce un array di LocationItem

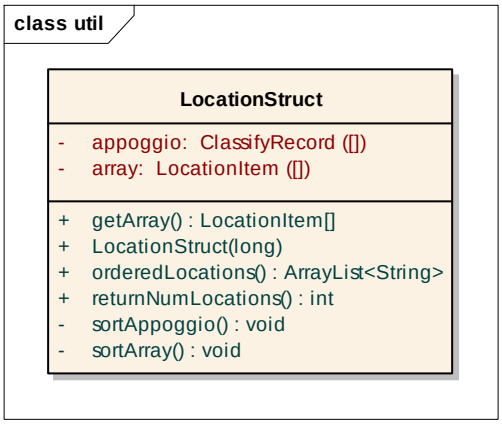

Figura 85 – LocationStruct

Metodi

Nome	Descrizione
LocationStruct(long)	Costruttore che realizza l'array di LocationItem per l'utente con ID dato in input
getArray(): LocationItem[]	Ritorna l'array di LocationItem creato ed ordinato
orderedLocations(): ArrayList<String>	Ritorna un'ArrayList delle locations ordinate
returnNum Locations(): int	Ritorna il numero di locations (grandezza dell'array)
sortAppoggio(): void	Ordina l'array di ClassifyRecord di appoggio
sortArray(): void	Ordina l'array di LocationItem

Tabella 58 - Metodi della classe LocationStruct

StringFunctions Realizza operazioni di manipolazione di stringhe.

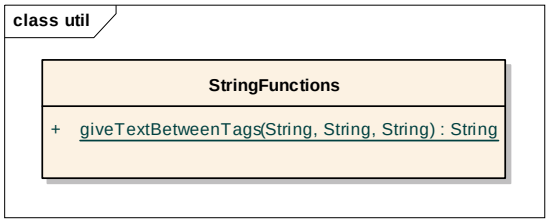

Figura 86 - StringFunctions

Metodi

Nome	Descrizione
giveTextBetweenTags (String, String, String): String	Ritorna il testo presente fra un tag di inizio e un tag di fine, data una stringa in input

Tabella 59 - Metodi della classe StringFunctions

4.6.2 Altri package

Per quanto riguarda gli altri package, non verrà mostrata la struttura delle classi, poiché essa consiste semplicemente delle seguenti caratteristiche:

- Ogni package realizza uno dei servizi previsti in CHAT;
- Ogni classe presente nel package realizza una delle funzioni viste ai paragrafi 4.4 – 4.5 e relativi sottoparagrafi.

5. Sperimentazioni

Alla fine della fase di implementazione del ciclo di vita di un'applicazione software avviene l'effettivo rilascio del sistema. Prima del rilascio, però, lo stesso modello a cascata dell'Ingegneria del Software prevede di effettuare una batteria di *test*, con la finalità di verificare che il sistema risponda a tutti i requisiti previsti, vale a dire sia che risponda alle specifiche ideate in fase di analisi e progettazione, sia che sia effettivamente fruibile dall'utente finale. [36]

Nel caso specifico di questo lavoro, però, lo scopo della sperimentazione non è quello di verificare la bontà specifica di ITR, bensì quello di comprendere se l'*utilizzo* di ITR all'interno di CHAT porti a un miglioramento sostanziale dei servizi offerti. Ciò significa che non devono essere valutati, per esempio, tempi e precisione del recommender ma, invece, si deve vedere, con qualche metrica specifica, se i servizi implementati migliorino o meno tramite ITR.

L'organizzazione della sessione di sperimentazione si articola sostanzialmente in tre fasi:

1. *Preparazione*, fissando la popolazione che dovrà portare a termine la sperimentazione e definendo i task;

2. *Esecuzione*, individuando la metrica migliore per valutare la prestazione dei servizi e come organizzare i dati per estrarre nel modo più efficiente i risultati;

3. *Valutazione*, quantificando, cioè, i risultati ottenuti per comprendere se essi sono soddisfacenti o meno.

5.1 Preparazione della sperimentazione

In questo paragrafo, dunque, verrà descritta la fase della sessione di sperimentazione denominata *preparazione*; la stessa si articola in tre passi:

1. Descrizione dell'*ambiente di lavoro*;
2. Descrizione della *popolazione;*
3. Descrizione del *task.*

5.1.1 Descrizione dell'ambiente di lavoro

Al fine di portare a termine il primo passo della fase di preparazione della sperimentazione è stata utilizzata una piattaforma realizzata *ad hoc* e disponibile online chiamata PinacotecaPHP. [37]

Figura 87 - PinacotecaPHP

Dunque, questa applicazione Web, scritta in PHP, è stata realizzata proprio per portare a termine la procedura di *training*. La pagina iniziale della applicazione indica gli obiettivi della stessa, le modalità di organizzazione e di raccolta dei dati. L'utente, dopo la registrazione, potrà votare le opere della Pinacoteca Vaticana con una scala da 1 a 5 ed etichettarla con un insieme di parole.

La scelta della realizzazione di questa applicazione ha permesso una facile fruizione della stessa da parte delle persone interessate: difatti, semplicemente tramite una connessione ad Internet e ad un browser che supporti PHP (allo stato attuale, quasi tutti), nonché ad un po' di tempo per completare il training, si è riusciti ad ottenere una buona quantità di dati.

5.1.2 Descrizione della popolazione

Un altro parametro importante per effettuare una buona valutazione è quello di scegliere in maniera adeguata la popolazione che effettuerà il training.

In primo luogo, bisogna comprendere la tipologia di utenti che utilizzerà i servizi e cercare di avvicinarsi quanto più possibile, in questa fase, ad essa. Secondariamente, è importantissima anche la numerosità del campione: è abbastanza scontato capire che, quanto più grande è il numero di individui che portano a termine la fase di training, tanto migliore sarà la qualità delle raccomandazioni, poiché quelle peggiori verrebbero isolate, facendo sì che la media delle stesse si portasse a livelli accettabili.

Dunque, come detto, data la reale facilità di utilizzo di PinacotecaPHP, si è potuto, in tempi abbastanza brevi, ottenere un buon campione di utenti, sia in termini di numero che in termini di eterogeneità.

Nel periodo di tempo che va dal 01/03 al 15/03, *trenta soggetti* hanno portato a termine l'addestramento. Essi si contraddistinguevano per essere in prevalenza di sesso maschile e di età inferiore ai 25 anni. Per quanto riguarda l'eterogeneità, si è potuto constatare che sia le competenze in ambito informatico (da laureati in informatica, ad appassionati di computer, a semplici fruitori dello stesso) che in ambito artistico (da intenditori di arte a classici visitatori di musei) erano quanto più variegate possibile. Questa varietà è corretta, dato che, ovviamente, la popolazione che visita musei rispecchia queste diversità.

5.1.3 Descrizione del task

Quest'ultimo passo della fase di preparazione riguarda la individuazione e descrizione dei task che dovranno essere effettuati dagli utenti oggetto dell'addestramento.

L'obiettivo della sperimentazione, come già detto in precedenza, è quello di valutare la bontà dei servizi di CHAT, se essi vengono migliorati con l'integrazione, all'interno di esso, di ITR. Dunque, gli utenti hanno dovuto portare a termine un task articolato in 4 fasi:

1. Iscrizione alla piattaforma;
2. Login;
3. Votazione delle opere;
4. Tagging delle opere.

Dopo l'iscrizione – *anonima*, dato che questa serve solo ad associare, a ciascun utente, un ID univoco utilizzato successivamente in fase di *test* – alla piattaforma, all'utente sono state mostrate le 45 opere da votare, suddivise in quattro blocchi da dieci ed un blocco da cinque, sempre estrapolate dal sito della Pinacoteca Vaticana.

A ciascuna opera, dunque, l'utente doveva associare un voto, compreso in una scala da 1 a 5, e un set di tag, o definibili in maniera autonoma o anche scelti a partire da quelli più utilizzati dagli altri utenti.

5.2 Esecuzione della sperimentazione

In questo paragrafo verrà descritta la seconda fase della sperimentazione, denominata *esecuzione* e composta anch'essa da 3 passi:

1. *Raccolta dei dati* da elaborare con ITR ed integrare in CHAT per valutare la bontà dei servizi;
2. *Definizione ed individuazione delle metriche* migliori per la valutazione delle prestazioni dei servizi;
3. *Organizzazione delle sessioni di sperimentazione*, per estrapolare i dati oggetto di valutazione.

5.2.1 Raccolta dei dati

Tutti i dati rilasciati dagli utenti, che sono in *rete*, dato che, come già detto, PinacotecaPHP è un'applicazione Web, sono stati memorizzati in modo persistente in un database MySQL e poi esportati in locale per la manipolazione degli stessi tramite ITR.

La procedura di raccolta dati prende il nome di *dump*: tecnicamente, questo consiste nel salvataggio dell'intero database in uno script .sql che poi si può facilmente importare in locale.

Come già accennato in precedenza, ciascuno dei 30 utenti ha votato un totale di 45 opere e le ha taggate con un insieme di parole. Tutti questi dati sono dunque stati esportati da PinacotecaPHP e successivamente importati in un computer in locale – nello specifico nel database di ITR – per poterli elaborare in una fase successiva.

5.2.2 Definizione delle metriche

L'individuazione delle metriche è il passo più importante per valutare in maniera efficiente e corretta i dati raccolti in fase di sperimentazione. Dunque, in questo passo si dovranno trovare dei parametri per valutare la qualità dei servizi offerti in CHAT, vale a dire per calcolare la bontà dei servizi se ad essi viene affiancato ITR.

La ricerca, nei sistemi di raccomandazione, utilizza diversi tipi di misure per valutarne la bontà. Nel nostro caso, dobbiamo considerare solo la qualità della predizione o della raccomandazione, dato che siamo interessati, per lo scopo della valutazione, solo all'output del nostro sistema di raccomandazione, ITR.

Dunque, si è deciso di adottare la *Normalized Distance-based Performance Measure (NDPM)* [38] per valutare la bontà dei ranking degli item calcolata in accordo ad una certa misura di relevance. Più specificatamente, *NDPM* viene utilizzata per misurare la distanza fra il voto dato agli item dai voti dell'utente (rappresentato da $>_u$) e il voto predetto dal sistema (rappresentato da $>_s$). Data una coppia di item (t_i, t_j)

nel Test Set *T* di un utente, si calcola la distanza fra di essi tramite il seguente schema:

$$\delta_{>u,\,>s}(t_i, t_j) = \begin{cases} 2 & \Leftrightarrow \quad (t_i >_u t_j \wedge t_j >_s t_i) \vee (t_i >_s t_j \wedge t_j >_u t_i) \\ 1 & \Leftrightarrow \quad (t_i >_s t_j \vee t_j >_s t_i) \wedge t_i \approx_u t_j \\ 0 & \Leftrightarrow \quad \textit{altrimenti} \end{cases}$$

Il valore dell'NDPM sul Test Set *T* è calcolato tramite la seguente equazione:

$$NDPM_{>u,>s}(T) = \frac{\sum_{i \neq j} \delta_{>u,>s}(t_i, t_j)}{2 \cdot n}$$

Dove *n* è il numero di coppie di item. I valori variano da 0 (agreement) ad 1 (disagreement).

Applicando la definizione di *NDPM* ai nostri servizi, potremmo utilizzarla nell'ambito della valutazione del servizio di *location*, dove il Test Set potrebbe essere una singola stanza, e dunque si possa calcolare il valore di agreement o disagreement per ciascun utente e ciascuna delle stanze prese in considerazione.

5.2.3 Organizzazione delle sessioni

Una volta individuata NDPM come metrica più adatta per lo specifico contesto di valutazione dei servizi, l'ultimo passo della fase di esecuzione della sperimentazione è quello di organizzare le sessioni.

Anche questo passo, ovviamente, risulta importante, dato che, per una migliore analisi statistica dei dati è necessario ed importante lavorare su differenti dataset, per evitare errori dovuti a valutazioni di un singolo dataset scelto magari in maniera poco corretta.

L'idea di partenza alla base della organizzazione delle sessioni è stata quella di scegliere stanze con il maggior numero di opere e dunque con il maggior numero di *item couples* oggetto della metrica *NDPM*. Sono state scelte dunque le stanze 12, 14 e 15 aventi, rispettivamente, 5, 4 e 5 opere.

Una volta scelte le stanze, si è deciso di agire nel seguente modo:

1. Costruzione di tre diversi *Training Set*, contenenti rispettivamente i voti relativi a 25, 20 e 15 opere;

2. Realizzazione, tramite *ITR*, dei profili di ciascuno dei trenta utenti per ognuno dei Training Set precedentemente costruiti (a partire, ovviamente, dai voti inizialmente raccolti tramite PinacotecaPHP);

3. Produzione degli *score* in fase di test, rispettivamente per le 20, 25 e 30 opere che non fanno parte del Training e salvataggio degli stessi nel database di ITR;

4. *Confronto* dei valori ottenuti tramite ITR con i voti dati dagli utenti in fase preliminare (e conservati in maniera persistente in un altro database) tramite NDPM.

5.3 Valutazione della sperimentazione

L'ultimo passo del processo di sperimentazione, denominato *valutazione*, serve ad utilizzare la metrica decisa in precedenza per estrarre informazioni utili per quantici fare la bontà dei servizi realizzati.

Prima di arrivare alla disamina dei risultati, però, è opportuno illustrare in maniera generale il processo di profilazione e classificazione attuata per calcolare le varie NDPM.

5.3.1 Profilazione e classificazione

Come detto, i dati registrati tramite PinacotecaPHP sono stati importati nel database di ITR ed utilizzati come dati di input.

Questi dati, come spiegato al paragrafo 5.2.3, sono stati manipolati per ottenere delle sessioni di test differenziate. Dunque, per ciascuno dei trenta utenti iscritti in fase di training, è stato manipolato il database originale realizzando tre diversi database, mantenendo, in ciascuno di essi, i rating relativi, rispettivamente, alle prime 15, 20 e 25 opere. Ciascuno di questi tre database è stato poi l'input di ITR per il calcolo degli score degli item facenti parte della sessione di test.

ITR, nello specifico, prevede tre tipologie di profili istanziabili:

- *Content-based*
- *Tag-based*
- *Content+Tag-based*

Il profilo *content-based*, come dice il nome stesso, si basa solo sul contenuto dell'item, non tenendo conto, cioè, dei tag degli utenti. Nel nostro scenario della Pinacoteca Vaticana, gli slot sono soltanto il titolo dell'opera (*title*), l'autore (*author*) e la descrizione (*description*).

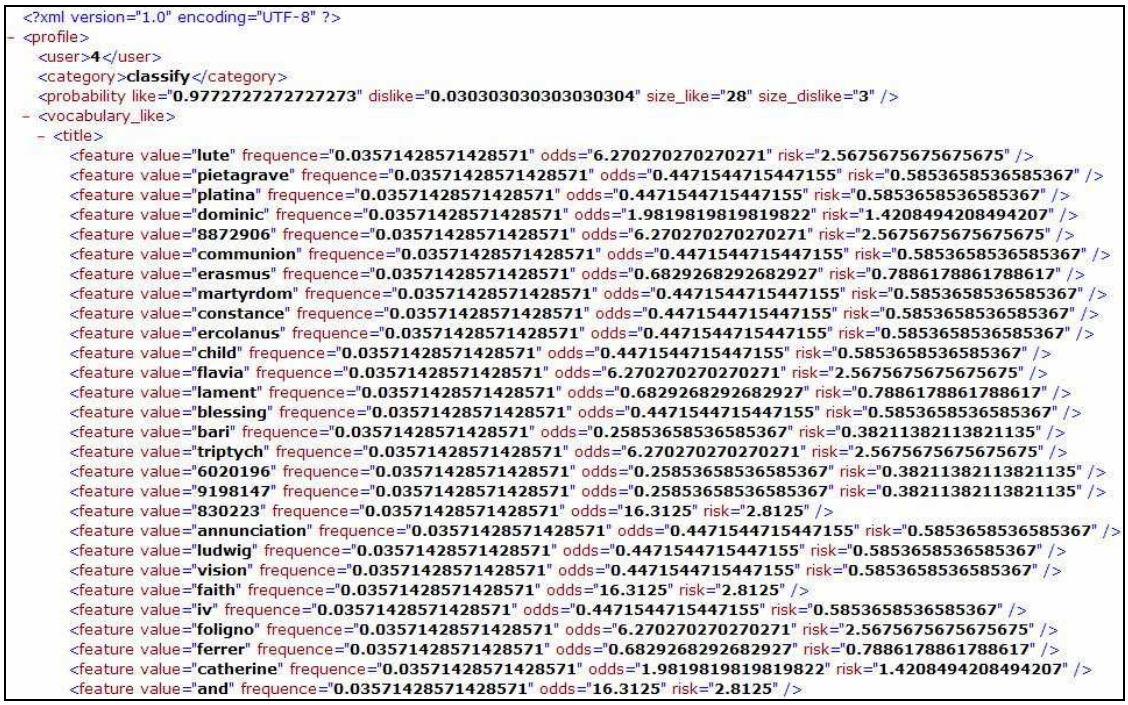

Figura 88 - Profilo content-based

Come si può vedere in figura, il file XML generato da ITR contiene le seguenti informazioni:

- ID dell'utente nel tag <*user*>;
- Score di preferenza per quella categoria nel tag <*probability*> tramite gli attributi *like* e *dislike*;
- Numero di elementi votati in modo positivo e negativo in fase di training nel tag <*probability*> tramite gli attributi *size_like* e *size_dislike*;
- I token che compongono gli item di gradimento per l'utente nel tag <*vocabulary_like*>;
- I token che compongono gli item non di gradimento per l'utente nel tag <*vocabulary_dislike*>;

- Per ciascun vocabolario, tre sottoelementi, <*title*>, <*author*> e <*description*>;

- Per ogni sottoelemento, un insieme di feature nel tag omonimo con attributi *frequence* (la frequenza normalizzata di quel token) *odds* e *risk* (indicatori di peso del token nel dataset).

Il profilo *tag-based* si basa solo sul contenuto generato dagli utenti, vale a dire dai tag degli stessi.


```
- <profile>
    <user>4</user>
    <category>Pittura</category>
    <probability like="0.6181818181818183" dislike="0.38181818181818183" size_like="6" size_dislike="3"/>
  - <vocabulary_like>
    - <tags>
        <feature value="xi" frequence="0.04651506202008269" odds="0.12280701754385964" risk="0.34210526315789475"/>
        <feature value="3376761" frequence="0.03307560137457045" odds="0.5357142857142858" risk="0.7936507936507937"/>
        <feature value="12776223" frequence="0.04282700421940928" odds="0.12280701754385964" risk="0.34210526315789475"/>
        <feature value="3474633" frequence="0.6852320675105484" odds="0.022556390977443608" risk="0.14473684210526316"/>
        <feature value="xv" frequence="0.039870311952330866" odds="2.6666666666666665" risk="1.6666666666666667"/>
        <feature value="7292449" frequence="0.8137130801687763" odds="0.12280701754385964" risk="0.34210526315789475"/>
        <feature value="7831551" frequence="0.14559594863447275" odds="11.111111111111111" risk="2.1111111111111111"/>
        <feature value="natura" frequence="0.03307560137457045" odds="3.571428571428571" risk="1.7142857142857142"/>
        <feature value="arte" frequence="0.07974062390466173" odds="2.6666666666666665" risk="1.6666666666666667"/>
        <feature value="2979974" frequence="0.04282700421940928" odds="0.12280701754385964" risk="0.34210526315789475"/>
        <feature value="bari" frequence="0.15948124780932346" odds="2.6666666666666665" risk="1.6666666666666667"/>
        <feature value="vii" frequence="0.04282700421940928" odds="0.12280701754385964" risk="0.34210526315789475"/>
        <feature value="3753656" frequence="0.07279797431723638" odds="3.571428571428571" risk="1.7142857142857142"/>
        <feature value="7748403" frequence="0.03307560137457045" odds="3.571428571428571" risk="1.7142857142857142"/>
        <feature value="6020196" frequence="0.3189624956186469" odds="2.6666666666666665" risk="1.6666666666666667"/>
        <feature value="7050461" frequence="0.0661512027491409" odds="3.571428571428571" risk="1.7142857142857142"/>
        <feature value="372083" frequence="0.03639898715861819" odds="3.571428571428571" risk="1.7142857142857142"/>
```


Figura 89 - Profilo tag-based

Nella figura sopra è possibile vedere che è scomparsa la struttura a tre sottoelementi – title, author e description – lasciando spazio soltanto ai tags che l'utente ha dato per le opere votate in fase di training.

È importante sottolineare che esistono due sottocategorie diverse per i profili tag-based:

- *Personal tags*, nel profilo confluiscono solo i tag dati dall'utente per le opere;
- *Social tags*, nel profilo confluiscono non solo i tag dati dall'utente per le opere, ma anche quelli che gli altri utenti hanno fornito.

Il profilo *content+tag-based*, infine, combina gli approcci content e tag based in un unico profilo.

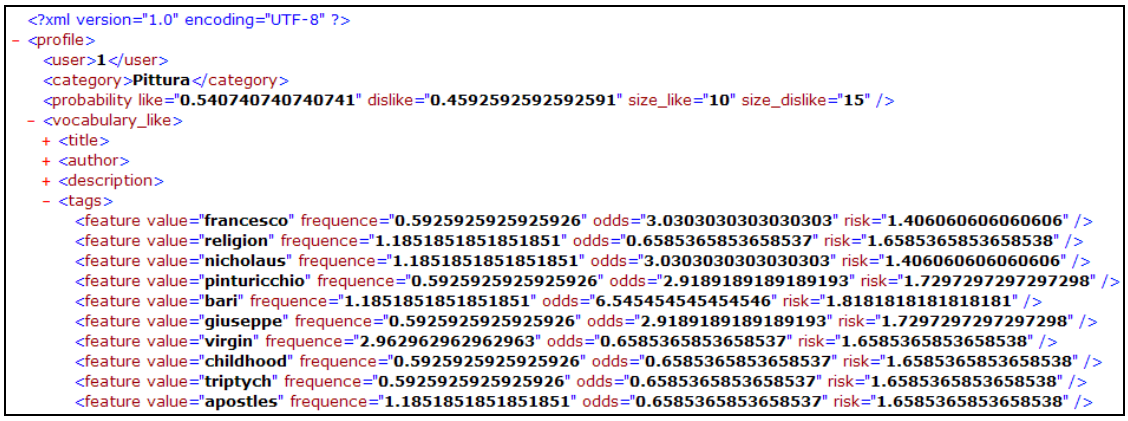

Figura 90 - Profilo content+tag-based

Come si può vedere in figura, questo profilo non è altro che un profilo content-based a cui è stato aggiunto un sottoelemento nei vocabolari like e dislike denominato <*tags*> e contenente, dunque, i tags per le opere votate, i quali possono essere *personal* o *social*, rispettando la differenziazione spiegata pocanzi.

È banale considerare il fatto che, quanto più grande è la quantità del contenuto da elaborare tramite ITR, tanto maggiore sarà il tempo di

realizzazione dei profili. Nel caso della realizzazione dei profili all'interno di CHAT, però, questo problema è di natura secondaria, dato che la batteria di profilazione e raccomandazione sarà mandata in esecuzione a run-time in periodo di scarso uso dell'applicazione (ad esempio, di notte) senza problemi sostanziali di prestazioni.

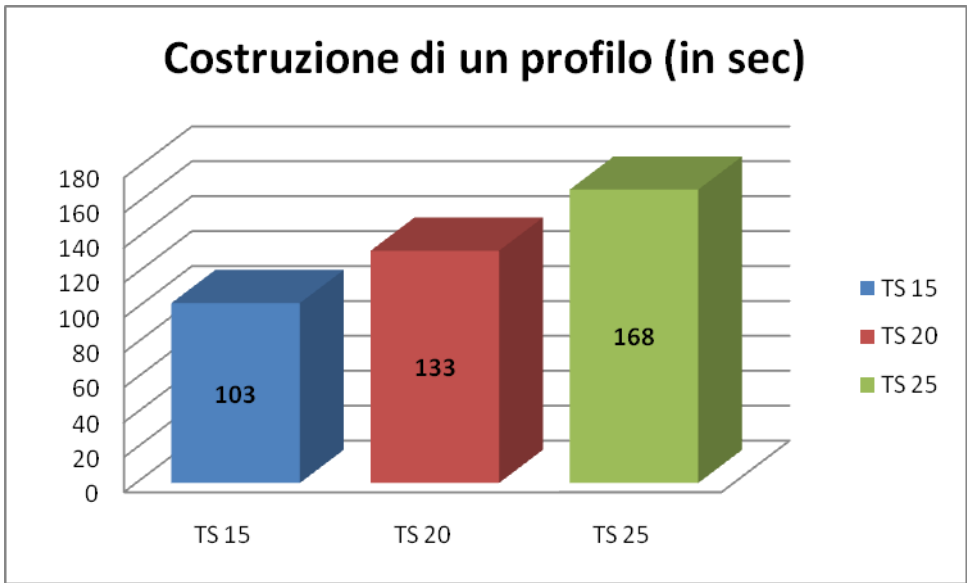

Figura 91 - Tempi di costruzione dei profili

La scelta tra le diverse tipologia di profilo è ricaduta sul profilo *content+tag-based* in versione *social tags*. I tempi in figura si riferiscono all'esecuzione di ITR su una macchina avente le seguenti caratteristiche:

- Processore Intel Core2 Duo T7300 2 GHz, 4 MB di cache livello 2;
- 2 GB di memoria RAM di tipo DDR2;
- 160 GB di Hard Disk a 5400 rpm.

La costruzione dei vari profili è l'ultima fase del processo di training. A questo punto, i voti dati alle opere e memorizzati in un database verranno

confrontati con quelli che il modello bayesiano di ITR stimerà sulla base dei tre dataset costruiti. Questa procedura prende il nome di *classificazione* e permette di individuare gli item di interesse per un particolare utente fra tutti quelli presenti nel Test Set.

L'elemento di interesse per la valutazione tramite NDPM è dunque lo score di appartenenza alla classe *like* per ciascun item del Test Set. Questo valore, infatti, sarà quello che andrà confrontato con i rating effettivi dati dall'utente agli item.

In ultimo, è importante precisare la differenza tra *risk* ed *odds*, valori che caratterizzano gli item per ciascun utente. Questi due valori altro non sono che pesi calcolati in fase di apprendimento dei profili, che tengono conto del voto espresso dall'utente su un item; durante la classificazione (tramite il file di configurazione di ITR) è poi possibile scegliere quale peso, tra odds e risk, utilizzare.

Una panoramica più approfondita e dettagliata sul funzionamento delle metriche è fornita in [39], ma nel nostro caso ci limiteremo a dire che il comportamento di *risk* riporta valori più alti in fase di classificazione, e dunque si è deciso di utilizzare questa metrica per le successive valutazioni.

5.3.2 Riepilogo dei risultati

In questo paragrafo saranno riassunti ed illustrati, in forma tabellare, tutti i dati dell'intera sperimentazione. Preliminarmente, è necessario indicare che la metrica NDPM è da considerarsi attendibile per valori minori di 0,4. Nel caso delle nostre sperimentazioni, però, dato che la sperimentazione ha coinvolto anche utenti non esperti e non è stata usata una grossa mole di dati, la soglia può essere spostata a 0,5.

Mostriamo, adesso, il riepilogo dei valori di NDPM per ciascun utente.

UTENTE 01

	R12	R14	R15	μ(TS)
TS15	0,40000	0,75000	0,50000	**0,55000**
TS20	0,65000	0,58333	0,50000	**0,57778**
TS25	0,55000	0,58333	0,50000	**0,54444**
μ(R)	**0,53333**	**0,63889**	**0,50000**	**0,55741**

Tabella 60 - Riepilogo per l'utente 01

UTENTE 02

	R12	R14	R15	μ(TS)
TS15	0,65000	0,41667	0,30000	**0,45556**
TS20	0,45000	0,58333	0,30000	**0,44444**
TS25	0,55000	0,58333	0,50000	**0,54444**
μ(R)	**0,55000**	**0,52778**	**0,36667**	**0,48148**

Tabella 61 - Riepilogo per l'utente 02

UTENTE 03

	R12	R14	R15	μ(TS)
TS15	0,50000	0,50000	0,80000	**0,60000**
TS20	0,40000	0,50000	0,70000	**0,53333**
TS25	0,20000	0,50000	0,70000	**0,46667**
μ(R)	**0,36667**	**0,50000**	**0,73333**	**0,53333**

Tabella 62 - Riepilogo per l'utente 03

UTENTE 04

	R12	R14	R15	μ(TS)
TS15	0,50000	0,66667	0,80000	**0,65556**
TS20	0,70000	0,50000	0,90000	**0,70000**
TS25	0,60000	0,50000	0,70000	**0,60000**
μ(R)	**0,60000**	**0,55556**	**0,80000**	**0,65185**

Tabella 63 - Riepilogo per l'utente 04

UTENTE 05

	R12	R14	R15	μ(TS)
TS15	0,60000	0,25000	0,50000	**0,45000**
TS20	0,70000	0,58333	0,60000	**0,62778**
TS25	0,70000	0,58333	0,60000	**0,62778**
μ(R)	**0,66667**	**0,47222**	**0,56667**	**0,56852**

Tabella 64 - Riepilogo per l'utente 05

UTENTE 06

	R12	R14	R15	μ(TS)
TS15	0,20000	0,83333	0,20000	**0,41111**
TS20	0,50000	0,83333	0,20000	**0,51111**
TS25	0,60000	0,83333	0,50000	**0,64444**
μ(R)	**0,43333**	**0,83333**	**0,30000**	**0,52222**

Tabella 65 - Riepilogo per l'utente 06

UTENTE 07

	R12	R14	R15	μ(TS)
TS15	0,40000	0,08333	0,30000	**0,26111**
TS20	0,60000	0,25000	0,40000	**0,41667**
TS25	0,70000	0,25000	0,40000	**0,45000**
μ(R)	**0,56667**	**0,19444**	**0,36667**	**0,37593**

Tabella 66 - Riepilogo per l'utente 07

UTENTE 08

	R12	R14	R15	μ(TS)
TS15	0,50000	0,75000	0,35000	**0,53333**
TS20	0,10000	0,75000	0,35000	**0,40000**
TS25	0,50000	0,75000	0,85000	**0,70000**
μ(R)	**0,36667**	**0,75000**	**0,51667**	**0,54444**

Tabella 67 - Riepilogo per l'utente 08

UTENTE 09

	R12	R14	R15	μ(TS)
TS15	0,60000	0,33333	0,30000	**0,41111**
TS20	0,45000	0,33333	0,30000	**0,36111**
TS25	0,45000	0,33333	0,40000	**0,39444**
μ(R)	**0,50000**	**0,33333**	**0,33333**	**0,38889**

Tabella 68 - Riepilogo per l'utente 09

UTENTE 10

	R12	R14	R15	μ(TS)
TS15	0,50000	0,58333	0,40000	**0,49444**
TS20	0,50000	0,25000	0,40000	**0,38333**
TS25	0,60000	0,08333	0,20000	**0,29444**
μ(R)	**0,53333**	**0,30556**	**0,33333**	**0,39074**

Tabella 69 - Riepilogo per l'utente 10

UTENTE 11

	R12	R14	R15	μ(TS)
TS15	0,50000	0,33333	0,60000	**0,47778**
TS20	0,55000	0,50000	0,40000	**0,48333**
TS25	0,55000	0,50000	0,20000	**0,41667**
μ(R)	**0,53333**	**0,44444**	**0,40000**	**0,45926**

Tabella 70 - Riepilogo per l'utente 11

UTENTE 12

	R12	R14	R15	μ(TS)
TS15	0,60000	0,50000	0,70000	**0,60000**
TS20	0,30000	0,50000	0,60000	**0,46667**
TS25	0,30000	0,50000	0,60000	**0,46667**
μ(R)	**0,40000**	**0,50000**	**0,63333**	**0,51111**

Tabella 71 - Riepilogo per l'utente 12

UTENTE 13

	R12	R14	R15	μ(TS)
TS15	0,40000	0,50000	0,40000	**0,43333**
TS20	0,50000	0,50000	0,60000	**0,53333**
TS25	0,40000	0,50000	0,60000	**0,50000**
μ(R)	**0,43333**	**0,50000**	**0,53333**	**0,48889**

Tabella 72 - Riepilogo per l'utente 13

UTENTE 14

	R12	R14	R15	μ(TS)
TS15	0,50000	0,50000	0,40000	**0,46667**
TS20	0,65000	0,50000	0,20000	**0,45000**
TS25	0,45000	0,50000	0,40000	**0,45000**
μ(R)	**0,53333**	**0,50000**	**0,33333**	**0,45556**

Tabella 73 - Riepilogo per l'utente 14

UTENTE 15

	R12	R14	R15	μ(TS)
TS15	0,60000	0,58333	0,15000	**0,44444**
TS20	0,15000	0,58333	0,25000	**0,32778**
TS25	0,25000	0,41667	0,15000	**0,27222**
μ(R)	**0,33333**	**0,52778**	**0,18333**	**0,34815**

Tabella 74 - Riepilogo per l'utente 15

UTENTE 16

	R12	R14	R15	μ(TS)
TS15	0,35000	0,33333	0,20000	**0,29444**
TS20	0,80000	0,50000	0,20000	**0,50000**
TS25	0,70000	0,50000	0,30000	**0,50000**
μ(R)	**0,61667**	**0,44444**	**0,23333**	**0,43148**

Tabella 75 - Riepilogo per l'utente 16

UTENTE 17

	R12	R14	R15	μ(TS)
TS15	0,80000	0,41667	0,30000	**0,50556**
TS20	0,60000	0,41667	0,30000	**0,43889**
TS25	0,70000	0,25000	0,30000	**0,41667**
μ(R)	**0,70000**	**0,36111**	**0,30000**	**0,45370**

Tabella 76 - Riepilogo per l'utente 17

UTENTE 18

	R12	R14	R15	μ(TS)
TS15	0,50000	0,25000	0,50000	**0,41667**
TS20	0,35000	0,25000	0,50000	**0,36667**
TS25	0,15000	0,25000	0,50000	**0,30000**
μ(R)	**0,33333**	**0,25000**	**0,50000**	**0,36111**

Tabella 77 - Riepilogo per l'utente 18

UTENTE 19

	R12	R14	R15	μ(TS)
TS15	0,50000	0,58333	0,40000	**0,49444**
TS20	0,30000	0,41667	0,50000	**0,40556**
TS25	0,40000	0,58333	0,50000	**0,49444**
μ(R)	**0,40000**	**0,52778**	**0,46667**	**0,46481**

Tabella 78 - Riepilogo per l'utente 19

UTENTE 20

	R12	R14	R15	μ(TS)
TS15	0,30000	0,25000	0,30000	**0,28333**
TS20	0,30000	0,41667	0,30000	**0,33889**
TS25	0,40000	0,58333	0,30000	**0,42778**
μ(R)	**0,33333**	**0,41667**	**0,30000**	**0,35000**

Tabella 79 - Riepilogo per l'utente 20

UTENTE 21

	R12	R14	R15	μ(TS)
TS15	0,65000	0,33333	0,55000	**0,51111**
TS20	0,40000	0,66667	0,55000	**0,53889**
TS25	0,40000	0,50000	0,55000	**0,48333**
μ(R)	**0,48333**	**0,50000**	**0,55000**	**0,51111**

Tabella 80 - Riepilogo per l'utente 21

UTENTE 22

	R12	R14	R15	μ(TS)
TS15	0,20000	0,50000	0,40000	**0,36667**
TS20	0,65000	0,50000	0,40000	**0,51667**
TS25	0,65000	0,50000	0,40000	**0,51667**
μ(R)	**0,50000**	**0,50000**	**0,40000**	**0,46667**

Tabella 81 - Riepilogo per l'utente 22

UTENTE 23

	R12	R14	R15	μ(TS)
TS15	0,45000	0,16667	0,50000	**0,37222**
TS20	0,40000	0,16667	0,50000	**0,35556**
TS25	0,50000	0,16667	0,70000	**0,45556**
μ(R)	**0,45000**	**0,16667**	**0,56667**	**0,39444**

Tabella 82 - Riepilogo per l'utente 23

UTENTE 24

	R12	R14	R15	μ(TS)
TS15	0,50000	0,50000	0,60000	**0,53333**
TS20	0,55000	0,66667	0,30000	**0,50556**
TS25	0,55000	0,66667	0,60000	**0,60556**
μ(R)	**0,53333**	**0,61111**	**0,50000**	**0,54815**

Tabella 83 - Riepilogo per l'utente 24

UTENTE 25

	R12	R14	R15	μ(TS)
TS15	0,40000	0,50000	0,30000	**0,40000**
TS20	0,15000	0,66667	0,30000	**0,37222**
TS25	0,15000	0,50000	0,30000	**0,31667**
μ(R)	**0,23333**	**0,55556**	**0,30000**	**0,36296**

Tabella 84 - Riepilogo per l'utente 25

UTENTE 26

	R12	R14	R15	μ(TS)
TS15	0,75000	0,25000	0,70000	**0,56667**
TS20	0,35000	0,25000	0,50000	**0,36667**
TS25	0,35000	0,25000	0,70000	**0,43333**
μ(R)	**0,48333**	**0,25000**	**0,63333**	**0,45556**

Tabella 85 - Riepilogo per l'utente 26

UTENTE 27

	R12	R14	R15	μ(TS)
TS15	0,60000	0,25000	0,45000	**0,43333**
TS20	0,30000	0,25000	0,55000	**0,36667**
TS25	0,50000	0,08333	0,55000	**0,37778**
μ(R)	**0,46667**	**0,19444**	**0,51667**	**0,39259**

Tabella 86 - Riepilogo per l'utente 27

UTENTE 28

	R12	R14	R15	μ(TS)
TS15	0,10000	0,25000	0,70000	**0,35000**
TS20	0,20000	0,41667	0,80000	**0,47222**
TS25	0,20000	0,41667	0,70000	**0,43889**
μ(R)	**0,16667**	**0,36111**	**0,73333**	**0,42037**

Tabella 87 - Riepilogo per l'utente 28

UTENTE 29

	R12	R14	R15	μ(TS)
TS15	0,60000	0,41667	0,70000	**0,57222**
TS20	0,70000	0,58333	0,60000	**0,62778**
TS25	0,60000	0,58333	0,60000	**0,59444**
μ(R)	**0,63333**	**0,52778**	**0,63333**	**0,59815**

Tabella 88 - Riepilogo per l'utente 29

UTENTE 30

	R12	R14	R15	μ(TS)
TS15	0,55000	0,41667	0,50000	**0,48889**
TS20	0,25000	0,75000	0,60000	**0,53333**
TS25	0,45000	0,75000	0,70000	**0,63333**
μ(R)	**0,41667**	**0,63889**	**0,60000**	**0,55185**

Tabella 89 - Riepilogo per l'utente 30

5.3.3 Curva di learning

Lo scopo della curva di learning è proprio quello di vedere la differenza nei risultati dell'applicazione di ITR a Training Set via via più piccoli. Si è deciso, come detto in precedenza, di creare tre TS da 25, 20 e 15 item e di utilizzare, poi, le stanze 12, 14 e 15 per valutare gli stessi in fase di Test.

In questo paragrafo, dunque, analizzeremo l'andamento delle curve per ciascuna stanza tramite opportuni grafici.

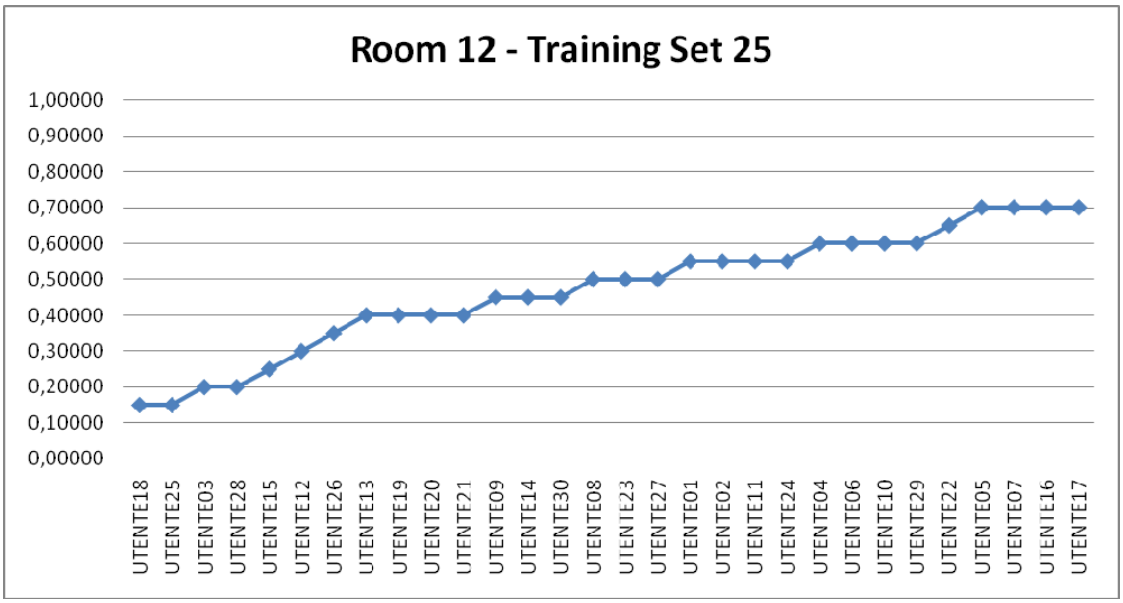

Figura 92 - Room 12 con Training Set 25

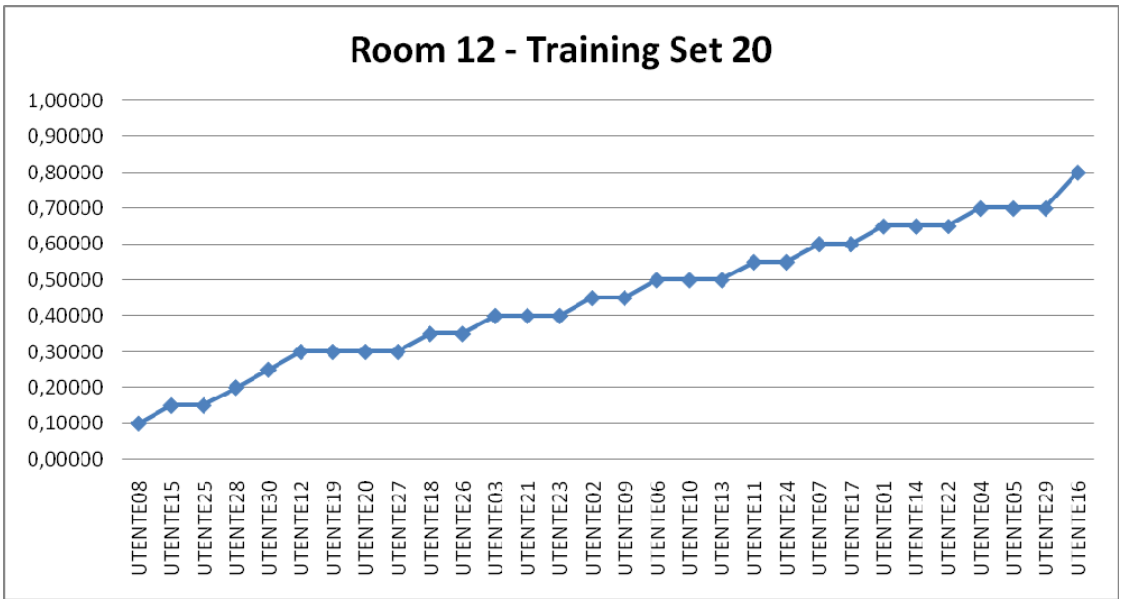

Figura 93 - Room 12 con Training Set 20

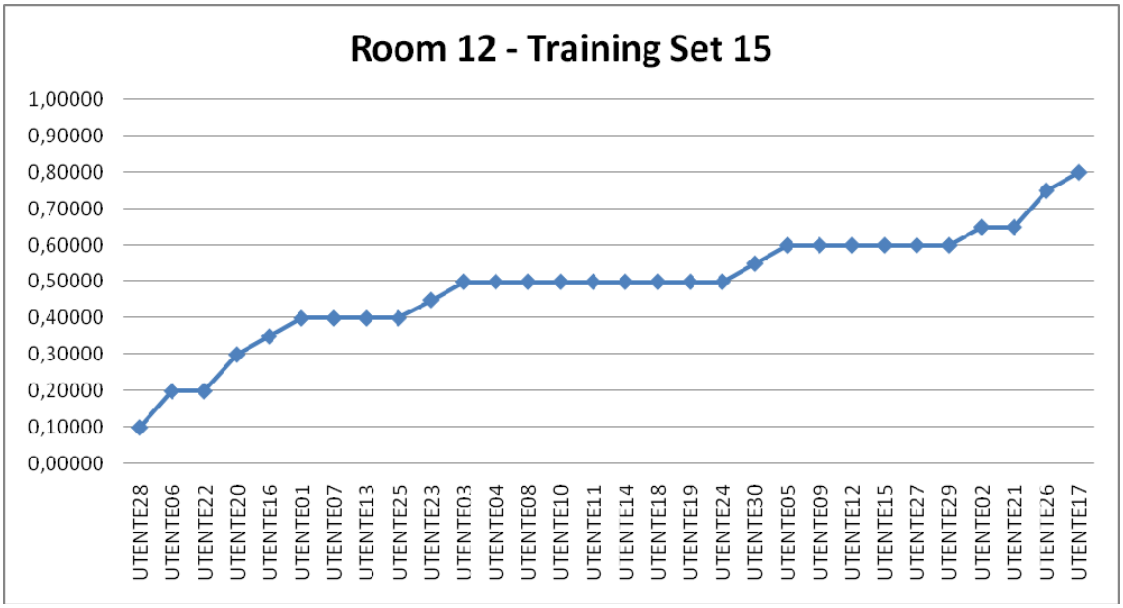

Figura 94 - Room 12 con Training Set 15

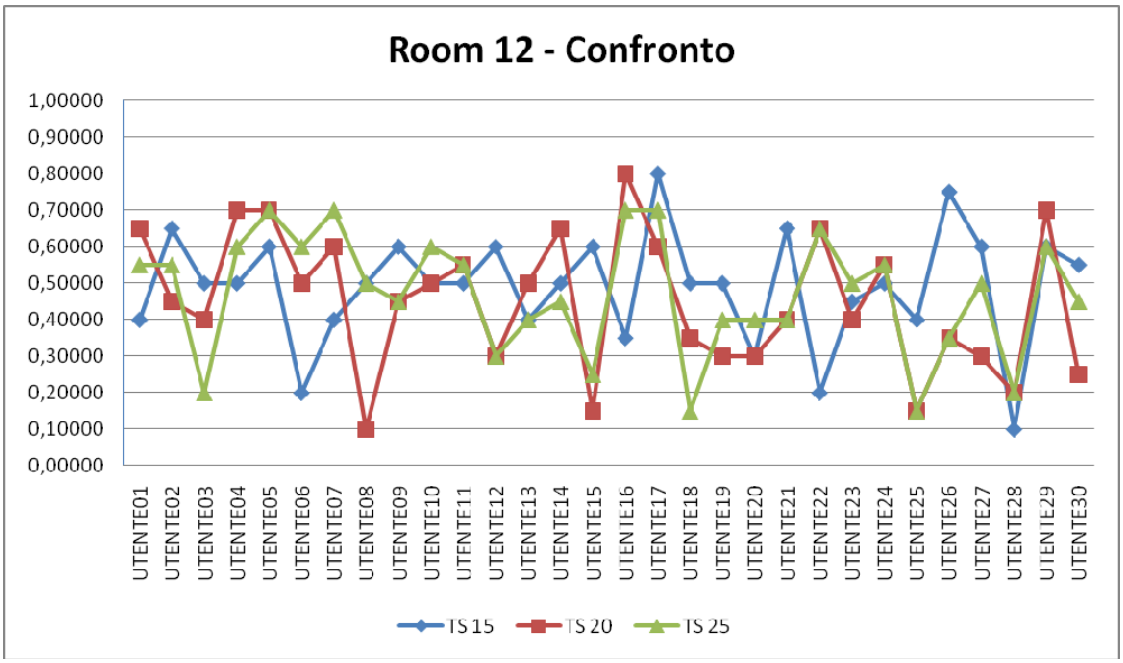

Figura 95 - Confronto tra i Training Set della room 12

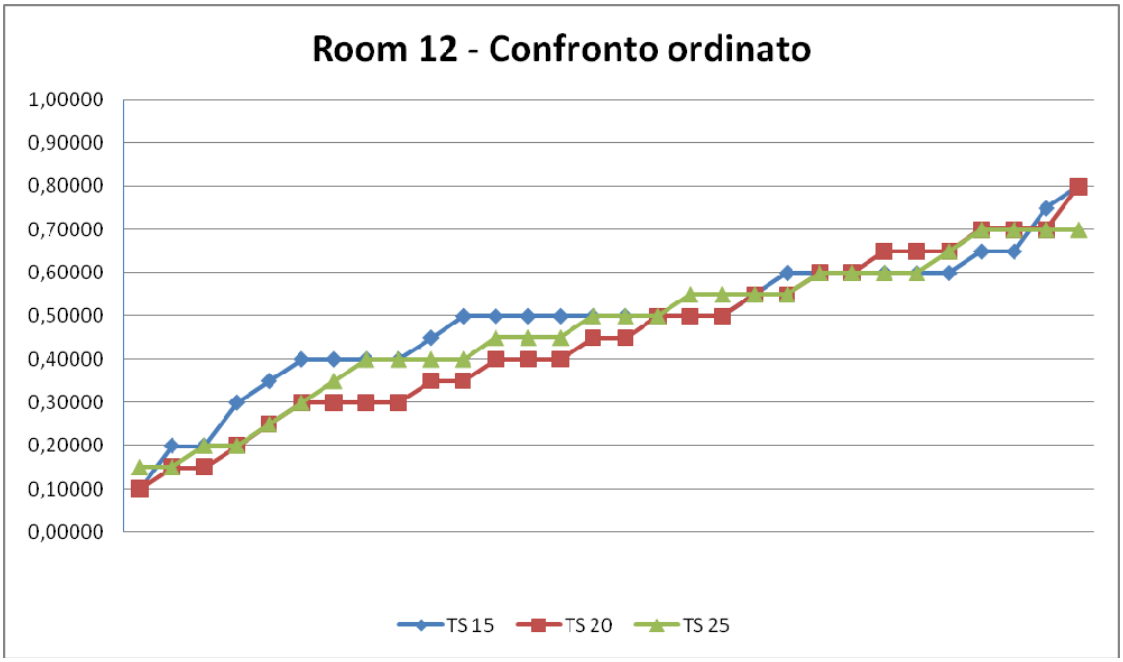

Figura 96 - Room 12, confronto con valori ordinati di NDPM

Per quanto riguarda la room 12, abbiamo come valori estremi 0,1 e 0,8. Possiamo vedere dalla figura 96 che, con un Training Set più piccolo, abbiamo anche dei picchi più alti.

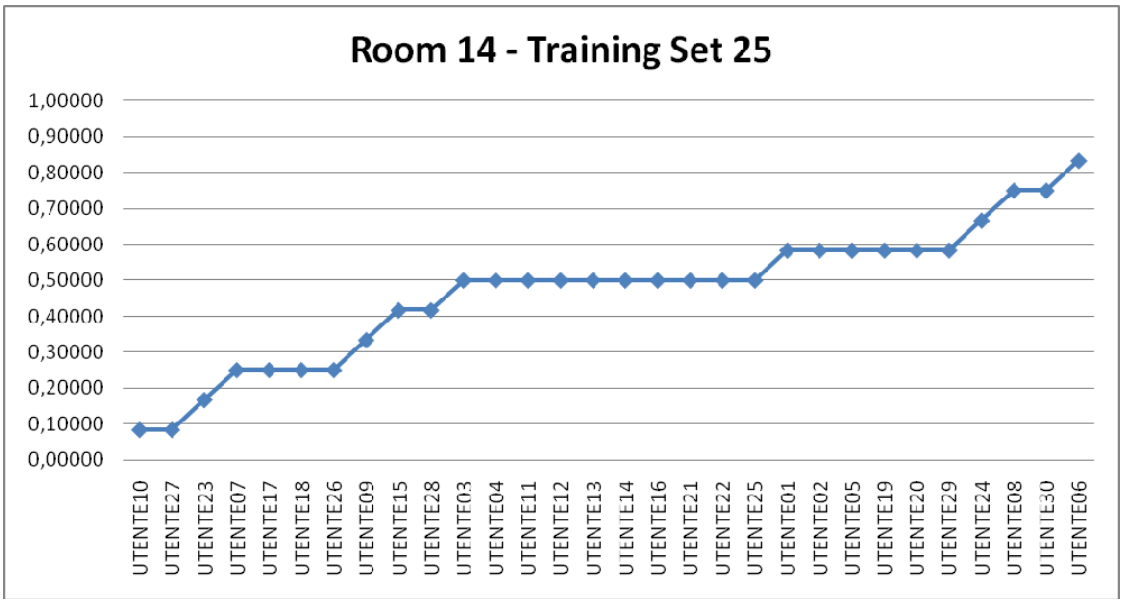

Figura 97 - Room 14 con Training Set 25

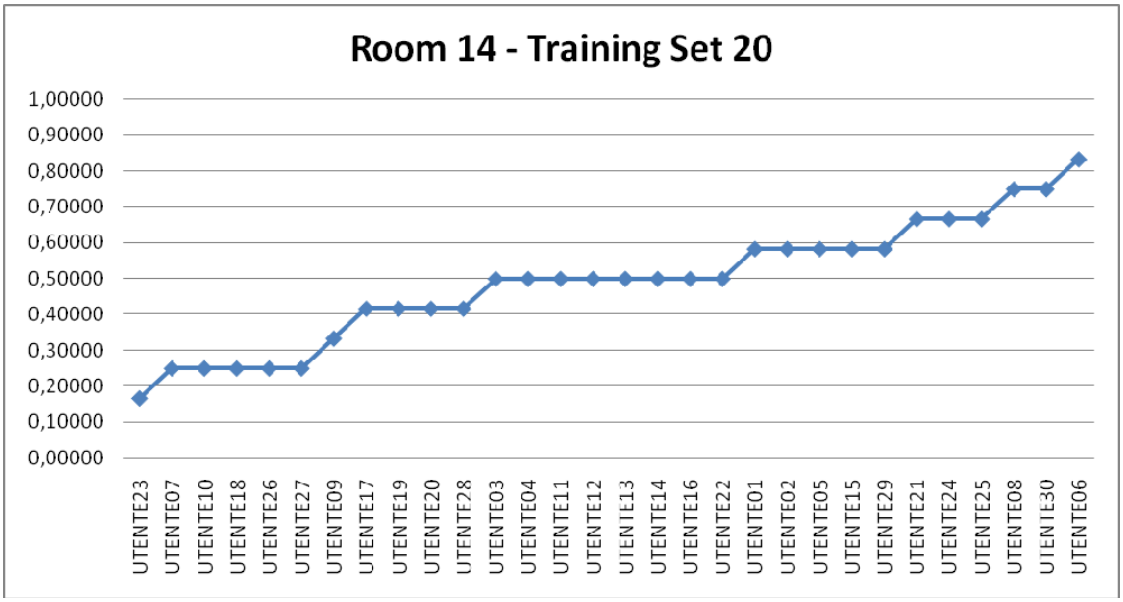

Figura 98 - Room 14 con Training Set 20

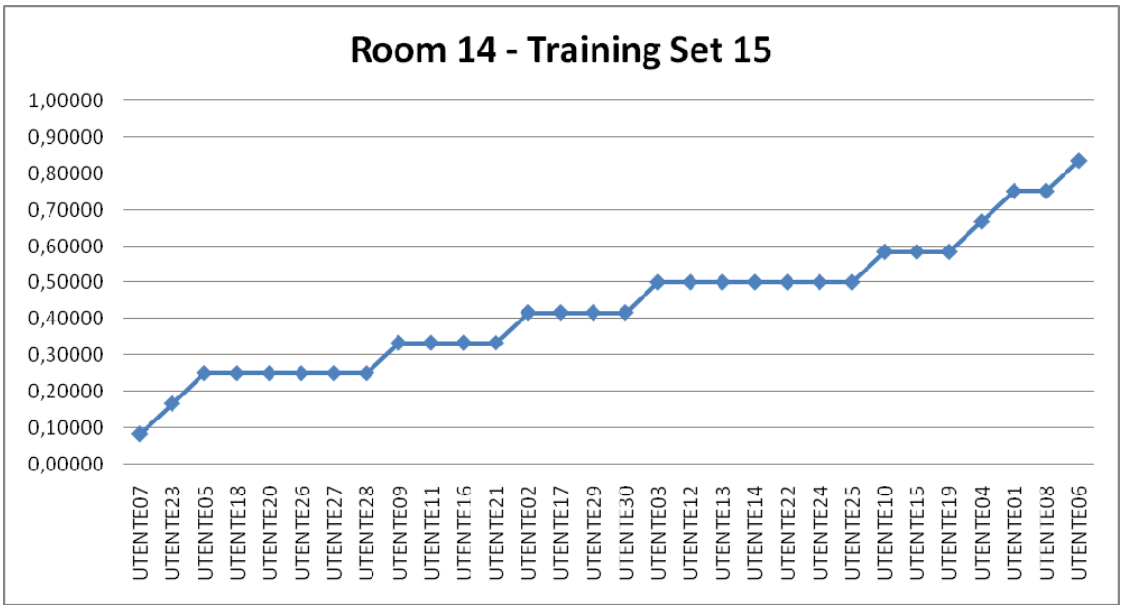

Figura 99 - Room 14 con Training Set 15

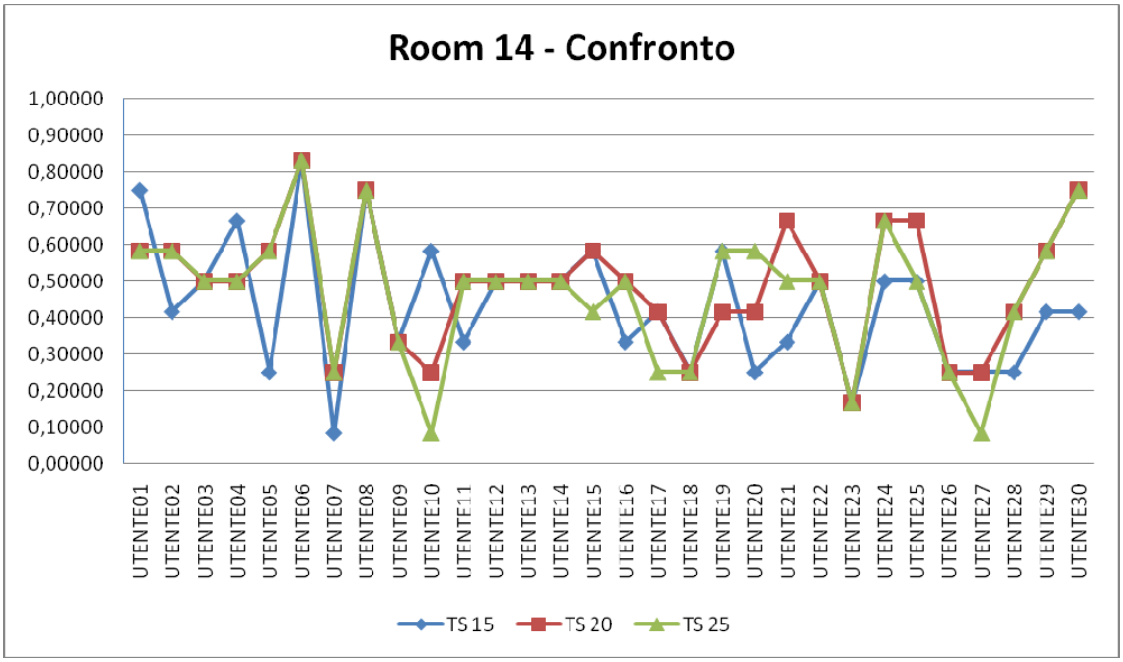

Figura 100 - Confronto tra i Training Set della room 14

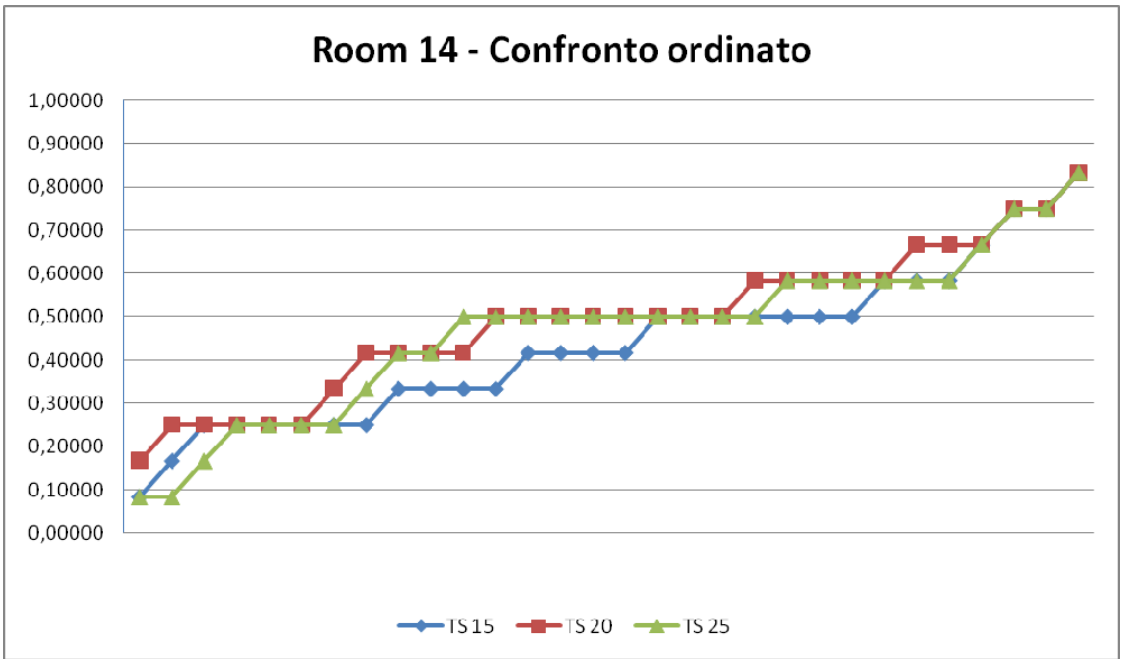

Figura 101 - Room 14, confronto con valori ordinati di NDPM

Per quanto riguarda la room 14, abbiamo come valori estremi 0,083 e 0,83. Possiamo intuire dalla figura 101 che la room è stata indubbiamente quella più difficile da classificare, ma sostanzialmente rispetta i canoni di decadimento delle prestazioni a seconda dei Training Set.

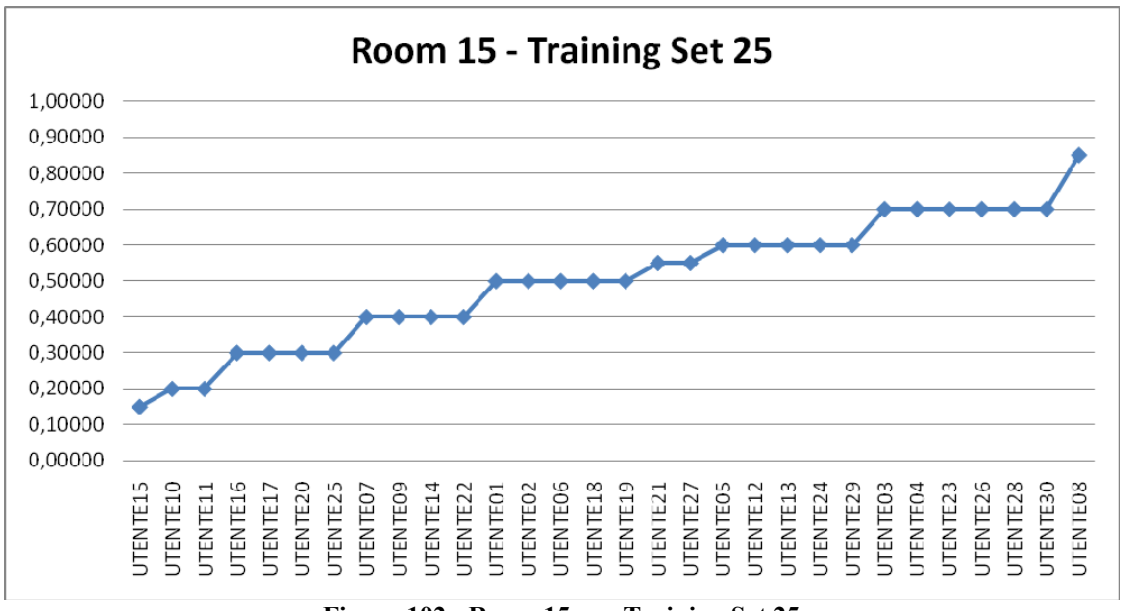

Figura 102 - Room 15 con Training Set 25

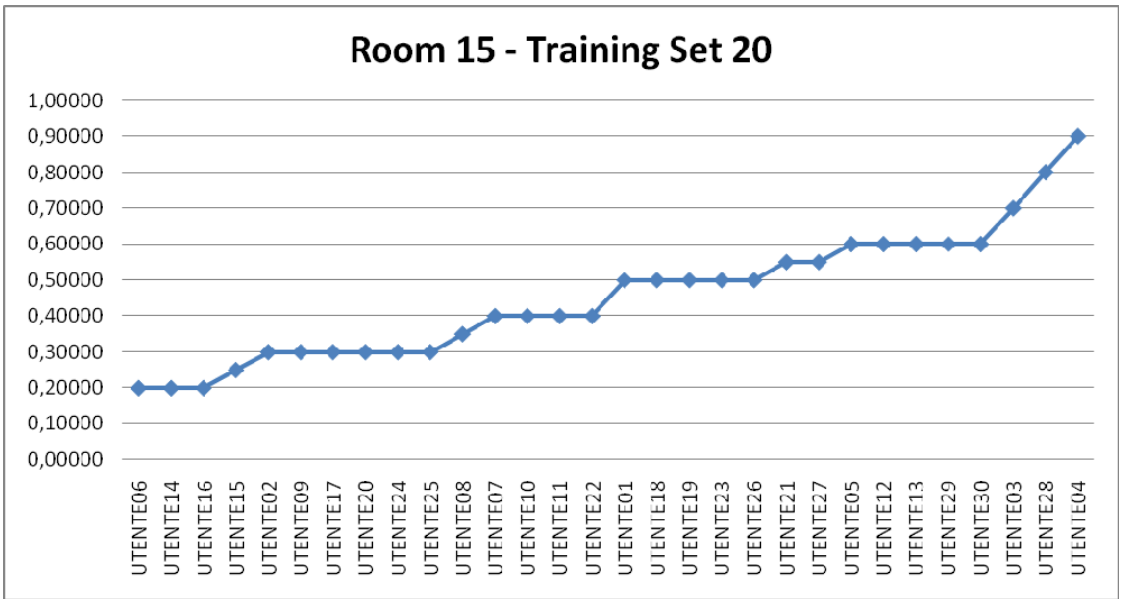

Figura 103 - Room 15 con Training Set 20

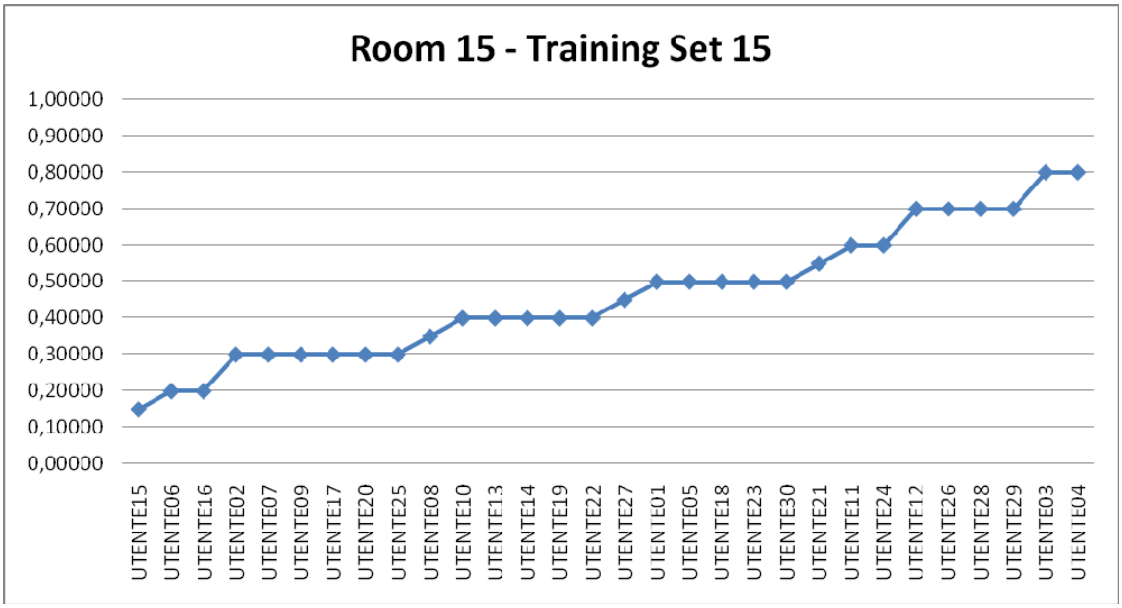

Figura 104 - Room 15 con Training Set 15

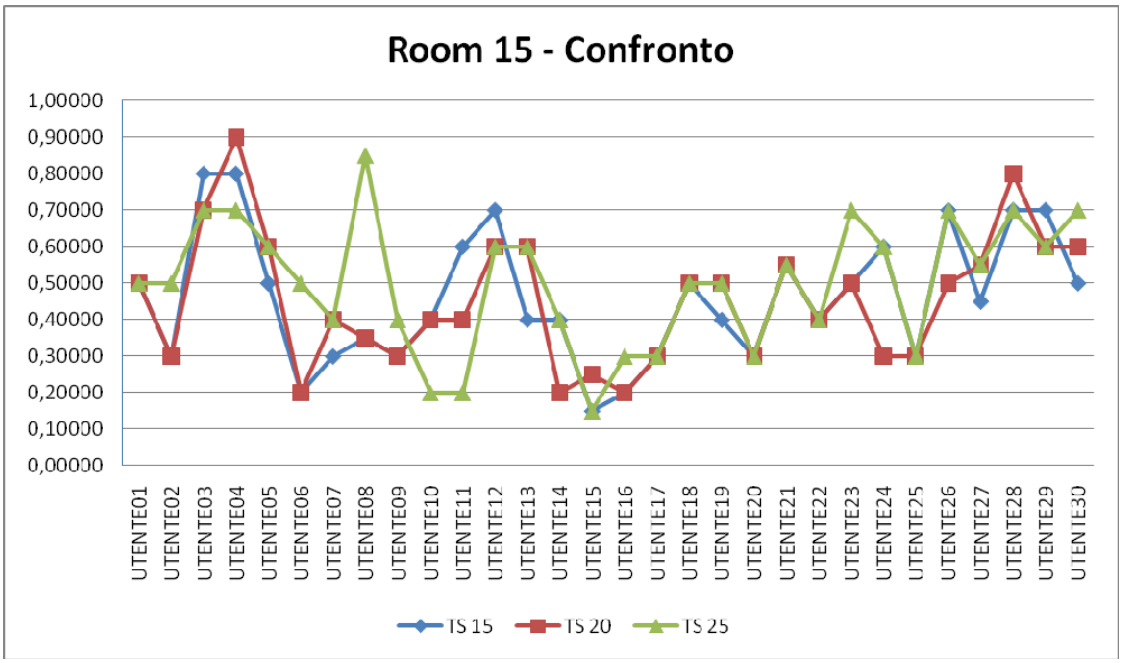

Figura 105 - Confronto tra i Training Set della room 15

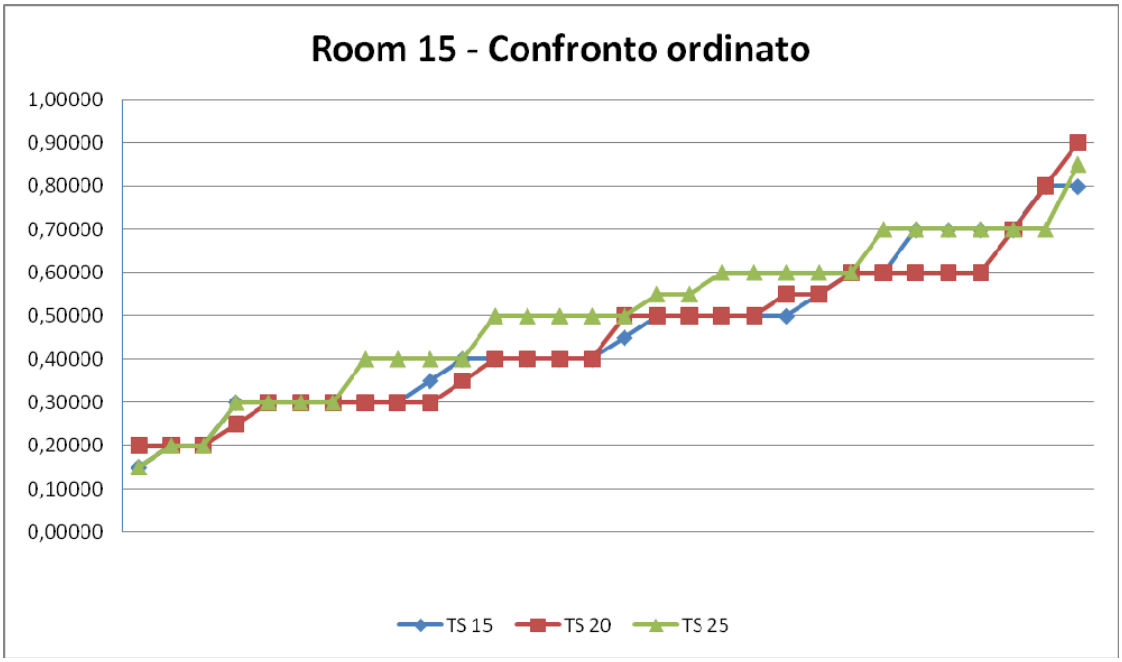

Figura 106 - Room 15, confronto con valori ordinati di NDPM

Per quanto riguarda la room 15, abbiamo come valori estremi 0,15 e 0,9. Guardando la figura 106, anche la room 15, come la 14, è difficile da classificare ma rispetta abbastanza bene il decadimento delle prestazioni a seconda dei Training Set.

5.3.4 Analisi dei risultati

In questo paragrafo verranno riassunti e visualizzati i risultati ottenuti da ITR se applicato nel contesto di CHAT, utilizzando NDPM come metrica.

Riepiloghiamo, adesso, le NDPM medie per ciascun utente:

RIEPILOGO UTENTI

UTENTE01	0,55741	**UTENTE16**	0,43148
UTENTE02	0,48148	**UTENTE17**	0,45370
UTENTE03	0,53333	**UTENTE18**	0,36111
UTENTE04	0,65185	**UTENTE19**	0,46481
UTENTE05	0,56852	**UTENTE20**	0,35000
UTENTE06	0,52222	**UTENTE21**	0,51111
UTENTE07	0,37593	**UTENTE22**	0,46667
UTENTE08	0,54444	**UTENTE23**	0,39444
UTENTE09	0,38889	**UTENTE24**	0,54815
UTENTE10	0,39074	**UTENTE25**	0,36296
UTENTE11	0,45926	**UTENTE26**	0,45556
UTENTE12	0,51111	**UTENTE27**	0,39259
UTENTE13	0,48889	**UTENTE28**	0,42037
UTENTE14	0,45556	**UTENTE29**	0,59815
UTENTE15	0,34815	**UTENTE30**	0,55185
		NDPM MEDIO:	**0,46802**

Tabella 90 - Riepilogo NDPM medie per utente

Come si può facilmente vedere, il valore della NDPM è inferiore alla soglia di 0,5 – dunque si può dire che il servizio location viene arricchito in maniera proficua dalle classificazioni di ITR. Inoltre, in ben 19 utenti su 30 il valore di NDPM è inferiore a 0,5.

Analizzando in maniera più approfondita questi dati, si può notare che ben 9 utenti si assestano sotto la soglia di 0,4 e altri 10 sotto quella di 0,5. Entrambi i dati sono abbastanza soddisfacenti, se si considerano i fattori riguardanti la numerosità del campione (30 persone), l'eterogeneità degli stessi, e, infine, la questione che alcuni degli utenti hanno votato le opere basandosi principalmente sull'immagine delle stesse e non soffermandosi

sulla descrizione testuale. Dato che ITR lavora, come sappiamo, sui token (o sui synset), ciò ha inficiato la classificazione degli item.

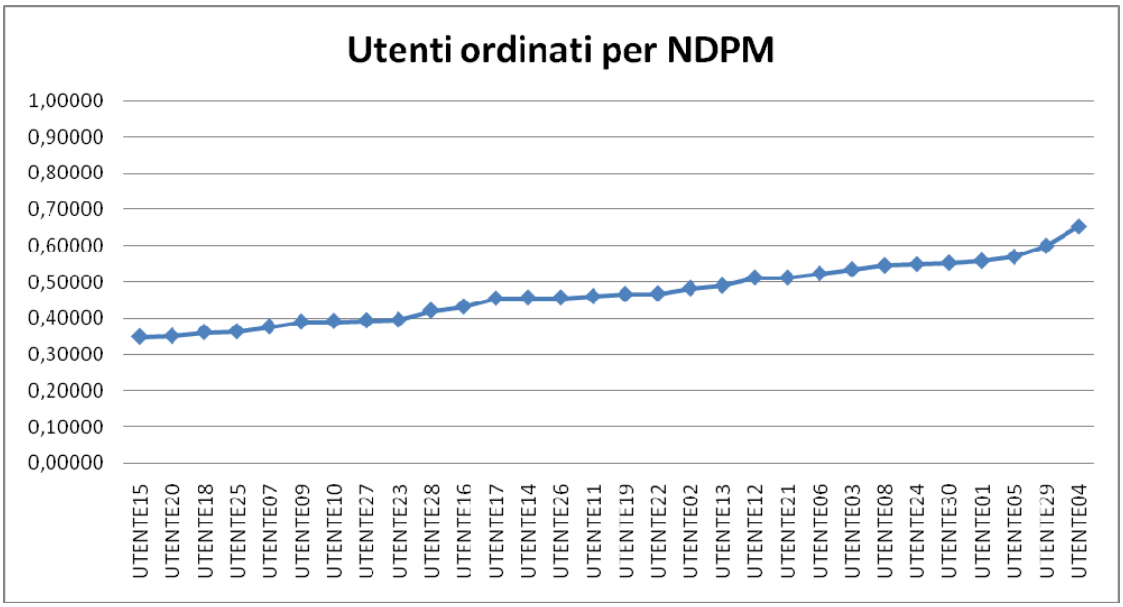

Figura 107 - Utenti ordinati per NDPM

Nella figura sopra, si può vedere come, a parte il picco finale dell'utente 04, c'è una crescita piuttosto lineare dei valori di NDPM, ma è altresì evidente che più della metà degli stessi si assestino sotto la soglia prestabilita.

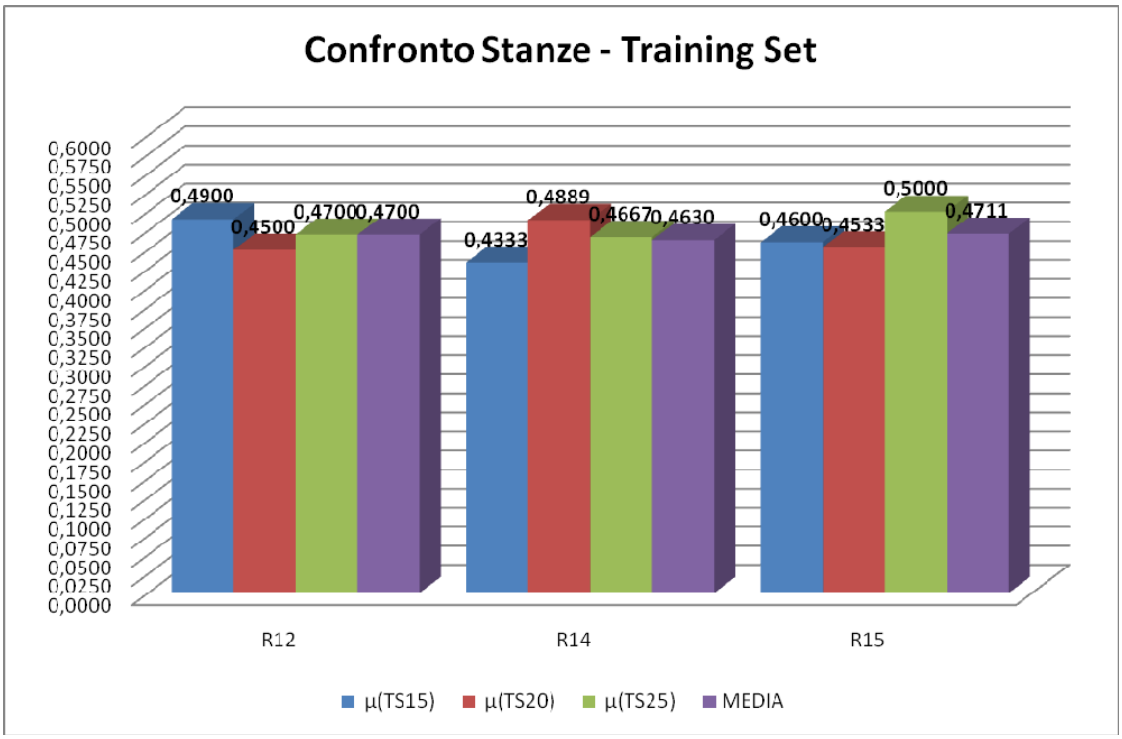

Figura 108 - Confronto fra le stanze classificate

In ultimo, la figura 108 indica l'andamento generale della NDPM media per ciascuna stanza. Come è possibile vedere, in alcuni casi la classificazione è risultata abbastanza difficile – con dei picchi che raggiungono la soglia di 0,5 – ma in generale i dati risultano in linea con le considerazioni precedenti.

6. Conclusioni e Sviluppi futuri

A conclusione di questo lavoro di tesi, e sfruttando quanto visto al capitolo precedente, è doveroso tirare le somme su quanto prodotto, sottolineandone punti di forza ed eventuali carenze da migliorare in futuro.

È indubbio che il punto di partenza di questo lavoro, la reingegnerizzazione di ITR, abbia portato un enorme vantaggio, sia per motivi di chiarezza a livello di codice, sia perché, adesso, la mappatura fra tabelle del database rende una futura manutenzione del pacchetto molto più semplice. Il recommender originale, infatti, prevedeva un'unica classe DbManager che conteneva tutte le funzioni di accesso al database, e la realizzazione dei profili era demandata a tre classi in un package che si richiamavano a vicenda.

Inoltre, l'altro evidente miglioramento su ITR è stato quello di svincolarsi dalla staticità degli slots: infatti, all'inizio la tabella Item del database era statica, legata agli item che andava a classificare, cioè i film. C'è da dire anche che c'era evidente bisogno di questa modifica per poter utilizzare ITR in qualsiasi contesto, e modificare la struttura del database in maniera semplice, da codice, tramite chiamate al metodo per la modifica del file di configurazione.

Per quanto riguarda i servizi realizzati in CHAT, essi hanno tratto giovamento proprio da questo nuovo recommender. Questi servizi, pensati per restituire informazioni statiche su opere, autori e location, sono stati arricchiti dalle classificazioni di ITR che permettono un filtraggio dei dati e la restituzione degli stessi adattati ad ogni singolo utente.

Il servizio location, oggetto poi delle sperimentazioni, è quello che maggiormente ha beneficiato degli score di probabilità calcolati tramite il

recommender. Difatti, anziché mostrare all'utente – come previsto nei capitolati del progetto CHAT – soltanto informazioni statiche relative agli ID degli oggetti presenti in una location, verranno mostrate informazioni aggiuntive quali stanze ordinate per importanza, item preferiti, ecc. Un utente, dunque, tramite queste informazioni potrà scegliere le location da visitare basandosi su dati che indicano quali sono quelle di suo gradimento.

I risultati delle sperimentazioni mostrati al capitolo 5 non fanno altro che supportare la tesi che questa integrazione di un recommender per l'utilizzo coi servizi di CHAT abbia portato benefici considerevoli. Analizzando le NDPM medie, infatti, abbiamo constatato che i valori sono accettabili, questo significa che i risultati della classificazione sono in linea con i feedback reali dati dagli utenti. Una persona interessata ad una visita museale, quindi, potrà "fidarsi" dei risultati forniti dai servizi implementati e visitare le opere più interessanti fra quelle profilate.

Gli sviluppi futuri prevedono alcune piccole migliorie che potrebbero permettere un utilizzo più efficace dei prodotti realizzati. Ad esempio, vista l'ampia disamina sullo stato dell'arte dei sistemi di raccomandazione al capitolo 1, una prima aggiunta potrebbe essere l'implementazione di un nuovo tipo di recommender, non di tipo bayesiano ma che sfrutti altre metriche fra quelle proposte. L'implementazione risulta anche abbastanza facile, dato che il Recommender è stato visto come un'*interfaccia* e dunque basterà rispettarne la struttura ed inserirlo nel package per essere praticamente utilizzato. In questo modo, anche le stime cambierebbero e si potrebbe valutare la migliore per utilizzare il recommender più adatto.

Per quanto riguarda i servizi, si potrebbe pensare all'utilizzo di una metrica migliore per la valutazione delle location. Infatti, quella utilizzata attualmente non fa altro che raggruppare le opere per location, generare lo

score medio per ciascuna ed ordinarle secondo due parametri: lo score medio stesso e il numero di opere presenti nella location (dando priorità alle location con il maggior numero di opere). In futuro sarebbe opportuno studiare misure statistiche più adeguate per misurare, ad esempio, la distanza fra due location.

In ultimo, è in cantiere la creazione di un “percorso personalizzato” per ciascun utente; un servizio, cioè, che indichi un tragitto fra le stanze di maggiore interesse per un utente. In verità, come sappiamo le location sono già rankate per importanza, e dunque un primo banale percorso sarebbe quello di attraversarle nello stesso ordine in cui vengono generate. Il miglioramento potrebbe essere fatto, ad esempio, sulla distanza fra le stanze, un parametro che attualmente non è presente in maniera persistente.

7. Ringraziamenti

Quando si arriva al momento dei ringraziamenti, siano questi la degna conclusione di un percorso di tesi, un semplice discorso da fare dopo un traguardo raggiunto, una parola da dire ad una persona a te cara, è bello che questi vengano dal cuore, è bello che le parole vengano fuori così, di botto…

<Ma no, Massimo… di botto? Ma che frase è "di botto"? Ma ti sembra questo il modo di scrivere? È pur sempre una tesi di laurea, questa… e per giunta di una tesi specialistica!>

…ok, torniamo seri. Ammesso che io ci riesca.

Primo Ottobre 2001 – Ventuno Aprile 2008. Come passa, il tempo! Duemilatrecentonovantacinque giorni, un numero che, scritto così in cifre, fa quasi paura. Ma noi informatici, si sa, dei numeri non dobbiamo avere alcun timore, allora sì, farò il forte come sempre, dicendo che, alla fine, ogni singolo giorno è stato un passo fondamentale nella mia vita, e non mi riferisco solo a quella universitaria.

Tanti avvenimenti duri, tanti problemi hanno in qualche maniera minato questo mio percorso. Non è facile arrivare a completare un percorso… ricordo ancora i voti "così così" nei primi esami, il baratro del secondo anno di università, che mi aveva quasi portato ad abbandonare, il tempo perso per questo, e poi la crisi post-laurea triennale, perché si sa, quando raggiungi il primo obiettivo, la voglia di continuare a studiare un po' ti passa.

Ma fra tutti i problemi, è meglio ricordare tutto ciò che invece mi ha dato la forza di andare avanti…

11/11/2003: è questa, senza dubbio, la data più importante della mia vita. C'è forse qualcosa più importante dell'amore? Non è forse l'amore quella forza che ti trascina, che ti fa superare tutti gli ostacoli? Quando sai di poter contare su una persona in qualsiasi momento, quando sai che lei è lì, ed anche se non è fisicamente vicina a te, basta prendere il telefono e parlare, perché tutto torni a posto?

Les enfants qui s'aiment s'embrassent debout	*I ragazzi che si amano si baciano in piedi*
Contre les portes de la nuit	*Contro le porte della notte*
Et les passants qui passent les désignent du doigt	*E i passanti che passano li segnano a dito*
Mais les enfants qui s'aiment	*Ma i ragazzi che si amano*
Ne sont là pour personne	*Non ci sono per nessuno*
Et c'est seulement leur ombre	*Ed è la loro ombra soltanto*
Qui tremble dans la nuit	*Che trema nella notte*
Excitant la rage des passants	*Stimolando la rabbia dei passanti*
Leur rage, leur mépris, leurs rires et leur envie	*La loro rabbia, il loro disprezzo, le risa e la loro invidia*
Les enfants qui s'aiment ne sont là pour personne	*I ragazzi che si amano non ci sono per nessuno*
Ils sont ailleurs bien plus loin que la nuit	*Essi sono altrove molto più lontano della notte*
Bien plus haut que le jour	*Molto più in alto del giorno*
Dans l'éblouissante clarté de leur premier amour	*Nell'abbagliante splendore del loro primo amore*

Jacques Prévert

Prévert è più bravo di me, io non sono bravo con le parole, lo sai Simona. Ma i ragazzi che si amano, non ci sono per nessuno… e io ti amo.

Adesso è difficile continuare coi ringraziamenti, ma ci provo. Iniziando con ciò che forse maggiormente mi ha aiutato negli studi, ma non solo. Quattromilaottocento utenti dal febbraio 2003, circa quattrocento visite giornaliere, è questo il biglietto da visita di *www.laureateci.it*

Nato un po' per gioco – mentre *facevamo finta di studiare* – si è fatto sempre più strada nell'ambiente universitario. E mi riempie di orgoglio dire che, nel gruppo di persone che hanno avuto questa idea, ci fossi anche io. *Chilavert*, è questo il mio *nickname*, ma è anche così che mi conoscono

molte delle persone iscritte al nostro Corso di Laurea. In effetti non so nemmeno se questo è un bene o un male, ma un po' di notorietà, secondo me, non ha mai fatto male a nessuno. E allora grazie, Laureateci, per aver creato quella piazza *virtuale* che ci ha fatto parlare, sorridere, ma anche arrabbiarci e litigare. Ma in fondo, che vita sarebbe se non ci fosse neanche una lite?

Un ringraziamento veramente speciale, un abbraccio grandissimo, alle due persone con cui ho condiviso questi due anni di Specialistica. Un grazie, dunque, a Cataldo e a Lucio, due persone così diverse fra loro ma accomunate da una grande qualità, che è ormai molto difficile da trovare. Sono due bravi ragazzi. Sembra poco? Per me non lo è. Grazie per tutto ciò che abbiamo fatto insieme. Ho tantissimi ricordi che mi vengono in mente in questo momento… come tutte le volte che ci siamo ammazzati di risate, in chat, la sera, parlando di cose così stupide che la mente umana probabilmente non riuscirebbe nemmeno a concepirle. O quando abbiamo partecipato alla conferenza, a fine Dicembre… un'esperienza veramente indimenticabile.

Come non ringraziare la mia famiglia, che mi ha supportato in tutti questi anni di studio, non soltanto economicamente – sebbene anche questa sia una componente fondamentale per andare avanti – ma anche supportandomi in alcuni momenti in cui le cose non andavano bene. Grazie a mio padre Giuseppe, a mia madre Lucia, a mia sorella Cinzia, a mio fratello Paolo ed a sua moglie Tania. E in ultimo, alla mia nipotina, Ester. Nei ringraziamenti della tesi di laurea triennale mi sono divertito a ricordare l'aneddoto del computer spento. Ester, infatti, ai tempi della laurea aveva soltanto due anni. Prendeva la rincorsa, e correndo spingeva il tasto dell'alimentazione del computer… e io perdevo un po' di codice che con tanta fatica avevo scritto. I tempi sono cambiati, Ester è cresciuta,

ma anche io ho prevenuto ogni problema. Ho deciso di studiare solo quando non era in casa!

Un grazie ai miei amici che mi hanno allietato in tutto questo tempo. Grazie ai miei compagni di università, a quelli che invece con l'università non c'entrano niente, a quelli che calcano i campi di calcetto con me… e a quelli che fanno parte dell'intersezione fra i vari insiemi, un doppio grazie!

E in ultimo, come non ringraziare i miei relatori… come non ringraziare chi ha dato fiducia al sottoscritto? E allora grazie al prof. Semeraro, a Pasquale Lops ed a Marco Degemmis, persone veramente squisite e che con me hanno avuto la massima disponibilità e gentilezza. Spero con tutto il mio cuore di aver dato a loro una buona impressione, e che abbiano apprezzato il mio impegno…

…e infine, grazie a me. Anche stavolta, ce l'ho fatta!

8. Bibliografia

[01] M. Balabanovic and Y. Shoham, “Fab: Content-Based, Collaborative Recommendation,” Comm. ACM, vol. 40, no. 3, pp. 66-72, 1997.

[02] M. Pazzani and D. Billsus, “Learning and Revising User Profiles: The Identification of Interesting Web Sites,” Machine Learning, vol. 27, pp. 313-331, 1997.

[03] G. Salton, Automatic Text Processing. Addison-Wesley, 1989.

[04] J.J. Rocchio, “Relevance Feedback in Information Retrieval,” SMART Retrieval System-Experiments in Automatic Document Processing, G. Salton, ed., chapter 14, Prentice Hall, 1971.

[05] Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. Addison-Wesley, 1999.

[06] G. Salton, Automatic Text Processing. Addison-Wesley, 1989.

[07] R.J. Mooney, P.N. Bennett, and L. Roy, “Book Recommending Using Text Categorization with Extracted Information,” Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08, 1998.

[08] G. Somlo and A. Howe, “Adaptive Lightweight Text Filtering,” Proc. Fourth Int’l Symp. Intelligent Data Analysis, 2001.

[09] Y. Zhang, J. Callan, and T. Minka, “Novelty and Redundancy Detection in Adaptive Filtering,” Proc. 25th Ann. Int’l ACM SIGIR Conf., pp. 81-88, 2002.

[10] S. Robertson and S. Walker, “Threshold Setting in Adaptive Filtering,” J. Documentation, vol. 56, pp. 312-331, 2000.

[11] Y. Zhang and J. Callan, “Maximum Likelihood Estimation for Filtering Thresholds,” Proc. 24th Ann. Int’l ACM SIGIR Conf., 2001.

[12] D. Billsus and M. Pazzani, “User Modeling for Adaptive News Access,” User Modeling and User-Adapted Interaction, vol. 10, nos. 2-3, pp. 147-180, 2000.

[13] E. Rich, “User Modeling via Stereotypes,” Cognitive Science, vol. 3, no. 4, pp. 329-354, 1979.

[14] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry,” Comm. ACM, vol. 35, no. 12, pp. 61-70, 1992.

[15] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl, “GroupLens: Applying Collaborative Filtering to Usenet News,” Comm. ACM, vol. 40, no. 3, pp. 77-87, 1997.

[16] J.S. Breese, D. Heckerman, and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” Proc. 14th Conf. Uncertainty in Artificial Intelligence, July 1998.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms,” Proc. 10th Int’l WWW Conf., 2001.

[18] D. Billsus and M. Pazzani, “Learning Collaborative Information Filters,” Proc. Int’l Conf. Machine Learning, 1998.

[19] L.H. Ungar and D.P. Foster, “Clustering Methods for Collaborative Filtering,” Proc. Recommender Systems, Papers from 1998 Workshop, Technical Report WS-98-08 1998.

[20] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, “Probabilistic Memory-Based Collaborative Filtering,” IEEE Trans. Knowledge and Data Eng., vol. 16, no. 1, pp. 56-69, Jan. 2004.

[21] A. Ansari, S. Essegaier, and R. Kohli, “Internet Recommendations Systems,” J. Marketing Research, pp. 363-375, Aug. 2000.

[22] Marco Degemmis, “Learning User profiles from text for personalized information access”, Jan 2005, Tesi di dottorato, Università degli Studi di Bari – Dip. Informatica

[23] Sauro Menchetti, “Estensione del Classificatore Naive Bayes per la Categorizzazione del Testo con Dati Parzialmente Etichettati” 1999-2000

[24] Bolt, R. A. (1980). Put-that-there: Voice and gesture at the graphics interface. Computer Graphics, 14 (3), 262-270.

[25] Oviatt, S.L., Cohen, P.R., Wu, L.,Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J. & Ferro, D. (2000). Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions. Human Computer Interaction, 15(4), 263-322. (reprinted in J. Carroll (Ed.) Human-Computer Interaction in the New Millennium, Addison-Wesley Press: Boston, 2001).

[26] Dupont, S., & Luettin, J. (2000, September). Audio-visual speech modeling for continuous speech recognition. IEEE Transactions on Multimedia, 2(3) 141-151. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

[27] Meier, U., Hürst, W. & Duchnowski, P. (1996). Adaptive bimodal sensor fusion for automatic speechreading. Proceedings of the International Conference on Acoustics, Speech and Signal Processing (IEEE-ICASSP), 833-836. IEEE Press.

[28] Rogozan, A. & Deglise, P. (1998). Adaptive fusion of acoustic and visual sources for automatic speech recognition. Speech Communication, 26(1-2), 149-161.

[29] Carpenter, R. (1992). The logic of typed feature structures. Cambridge, U. K.: Cambridge University Press.

[30] Wu, L, Oviatt, L, Cohen, P R. From members to teams to committee: A robust approach to gestural and multimodal recognition. IEEE Trans. On Neural Networks vol 13, no 4 July 2002.

[31] Pinacoteca dei Musei Vaticani online, http://mv.vatican.va/2_IT/pages/PIN/PIN_Main.html

[32] Restlet – Lightweight REST Framework for Java, http://www.restlet.org.

[33] C. Musto & F. Narducci. PinacotecaRest, Università degli Studi di Bari, Dipartimento di Informatica, 2008.

[34] OpenKapow – Mashup in minutes, http://openkapow.com/.

[35] C. Musto & F. Narducci. Pinacoteca2Fedora, Università degli Studi di Bari, Dipartimento di Informatica, 2008.

[36] ISO 12207 and Related Software Life-Cycle Standards – http://www.acm.org/tsc/lifecycle.html

[37] C. Musto & F. Narducci. PinacotecaPHP, Università degli Studi di Bari, Dipartimento di Informatica, 2008 - http://cataldo.netsons.org/tesi

[38] Yao, Y. Y. (1995) Measuring Retrieval Effectiveness Based on User Preference of Documents. In Journal of the American Society for Information Science, 46 (2), 133–145.

[39] F. Narducci. Stima dei parametri e feature weighting in un classificatore bayesiano: applicazione della distribuzione di Poisson e delle misure di rischio, Università degli Studi di Bari, Dipartimento di Informatica, 2008 (Tesi di Laurea Specialistica).