

Sistemi intelligenti per internet

by Marco Giannone e Marco Bontempi
(A.A. 2009-2010)

1. Seeking	3
1. Definizione di dato:	3
2. Tipologie di sistemi software:	4
3. Information Overload (sovraccarico di informazioni).....	4
4. Information Needs	5
5. Information Seeking	6
1. Information Filtering	8
2. Information Retrieval	9
3. Information Filtering VS Information Retrieval	10
6. Information Behaviour	11
2. Text Classification	12
1. Indicizzazione automatica per sistemi di IR.....	12
2. Approccio Knowledge Engineering	12
3. Approccio Machine Learning.....	13
1. Indicizzazione dei documenti e riduzione dimensionale:.....	13
2. Induzione di classificatori:.....	14
3. Valutare un classificatore di testi.....	19
3. Web Document Modeling LSI SVM	20
1. Latent Semantic Indexing.....	20
2. SVD	21
3. SVD in LSI in conclusioni.....	22
4. Introduzione allo User Modeling	24
1. Modellare la conoscenza	25
2. Modellare Goal e tasks	28
1. Modellare il Background	29
2. Modellare caratteristiche peculiari dell'utente	30
3. Modellare il contesto di lavoro	30
3. Impiego della componente UM nei sistemi di ricerca	30
5. Dispensa 7 Tecnologie per la Modellazione Utente	31
1. Recupero dei dati sull'utente	33
1. Explicit user feedback:	33
2. Implicit feedback	33
6. Tecnologie per la modellazione dell'Utente	35
1. Inizializzazione del modello Utente	35
7. Content-based Filtering, Collaborative Filtering, Focused Crawling (visualizzazione delle informazioni)	39
1. Content-based Filtering:	39
2. Collaborative Filtering:.....	39
3. Focused Crawling.....	40

Seeking

Internet come fonte di informazione: rende disponibile una grossa mole di dati.

Due problemi:

- Le dimensioni di internet sono molto vaste
- Capire cosa si sta cercando, spesso l'utente quando va su un motore di ricerca non sa cosa sta cercando.

Uno degli obiettivi dei Sistemi per Internet è quello di sfruttare tecniche di Intelligenza Artificiale per accedere "intelligentemente" ad Internet (vedi i-Access), cioè:

- Ridurre il tempo per ottenere informazioni
- Ridurre le risorse cognitive impiegate durante la ricerca di informazioni
- Suggerire informazioni più attinenti alle attività (task) correnti

Fonti di Informazioni su Internet (Information Services):

- Motori di ricerca
- Biblioteche elettroniche
- Banche dati
- Forums/Blogs

Deep Web: Parte del web non indicizzabile da un motore di ricerca...non indicizzabile perchè stanno dietro a dei box di ricerca (il Deep Web è pari a 550 volte la dimensione della parte indicizzabile).

Abbiamo una dimensione molto grande, gli utenti con il browser non ce la fanno e alcuni strumenti non sono adatti.

Dovremmo pensare ad altri strumenti, ci interessa approfondire i processi di information seeking per aggiungere dei blocchi ai servizi per individuare il bisogno informativo dell'utente.

Definizione di dato:

Qualcosa che puo' essere rappresentato dal nostro dominio.

Sono alla base del processo di ragionamento.

L'informazione è il risultato del processamento.

Facendo una certa analisi su dei dati in output otteniamo informazione.

Se non interviene conoscenza né un processo vero e proprio il dato rimane dato.

L'informazione può essere vista perciò come una collezione di dati da cui una persona può ricavare certe conclusioni.

L'informazione la trasmettiamo facilmente...la conoscenza viene acquisita tramite il ragionamento.

Quindi vediamo l'informazione come: *L'atto di organizzare e manipolare un insieme di dati in modo tale da variare la conoscenza della persona che la riceve.*

Gerarchia DIKW(data, information, knowledge, wisdom): modello generico per rappresentare le relazioni funzionali tra dati, informazione, conoscenza e saggezza (wisdomà sistemi esperti).

Tipologie di sistemi software:

Information System: Sistemi software impiegati nei processi di memorizzazione e recupero di dati. Suddividiamo l'information system in alcuni sottoprocessi:

- **ricerca**
- **recupero** (quando ho una base di informazione statica e devo recuperare dati da questa base che abbiamo già costruito)
- **filtraggio** (per flussi di informazione)

I task non vengono quasi mai rappresentati, siamo consapevoli noi del nostro task, ma il motore di ricerca non ne sa nulla.

Knowledge Management System: Si focalizzano sul Know how (come operare) relativo a processi più o meno articolati o attività ripetitive; esperienza acquisita mediante attività passate. Utilizzati per sostenere processi e task knowledge-intensive. Oltre a rappresentare informazione esplicita (documenti), i KMS costruiscono una base di conoscenza a partire dal comportamento degli individui, monitoraggio di colloqui chat, mail... *I KMS monitorano canali di comunicazione e fanno inferenza per interpretare situazioni, attività, comportamenti e soluzioni.*

Obiettivi:

- Catturare, creare e condividere i best practices (tecniche, metodi e processi ottimali per una certa attività/scopo)
- Essere di supporto a sistemi di Experience Management e Decision Support
- Creare tassonomie che organizzano e mettono in relazioni elementi del mondo
- Individuare skills e esperti di settore
- Creare communities o knowledge networks dove poter condividere conoscenza interazioni tra utenti

Otteniamo: performance, competitività, innovazione, condivisione della conoscenza.

Information Overload (sovraccarico di informazioni)

Fenomeno che si manifesta quando l'utente interagisce con l'informazione.

Il problema è nel modo di gestire le informazioni, nessuno va contro l'eccesso, il problema è allocare efficacemente l'attenzione e gestire l'informazione stessa.

Un eccesso di informazione crea una deficienza nell'attenzione. Per questo occorre allocare efficacemente l'attenzione rispetto alle sorgenti disponibili.

E' lo stato in cui un utente (o anche un sistema) non è in grado di processare o filtrare la mole di dati in input. Avviene tipicamente un fallimento nelle normali funzionalità.

7 Comportamenti che l'utente intraprende per affrontare l'Information Overload:

Disfunzionali:

- *Omission*: evitare di processare alcuni input.
- *Error*: processare alcuni input in modo non corretto
- *Escaping*: evitare di processare gli input ed interrompere le attività correnti

Potenzialmente disfunzionali:

- *Queueing*: ritardare il processamento di alcuni input
- *Approximation*: ridurre gli standard di precisione nel processare/categorizzare gli input.

- *Filtering*: processare solo gli input giudicati di maggiore interesse
- *Multiple Channels*: suddividere il flusso di informazioni in modo da decentralizzare il processamento

Information Needs

Un bisogno informativo nasce quando un individuo sente che la propria conoscenza è inadeguata (in riferimento ad un compito/obiettivo che ha). Un buco nella conoscenza sprona l'utente a cercare informazione.

Modello ASK:

Concezione del mondo che circonda l'utente; consapevolezza della propria conoscenza e delle lacune è stato anomalo della conoscenza è spinto a richiedere delle informazioni. Altri individui creano informazioni con concezioni della realtà e livelli di conoscenza simili o distinti con l'utente. Inoltre chi ha prodotto l'informazione può aver avuto determinate credenze e scopi che lo hanno spinto a pubblicizzarla. Lo scambio di conoscenza è mediato dai livelli di conoscenza tra individui e dal mondo fisico che ci circonda.

Bisogno informativo per raggiungere un determinato goal o task, un bisogno informativo fine a se stesso non lo consideriamo...

A volte però l'utente è inconsapevole del proprio bisogno informativo, si ha un grado di conoscenza scarso del dominio di interesse corrente e l'individuo non sa neppure cosa cercare esattamente oppure quali strumenti e fonti consultare.

(IMPORTANTE)

Stati del bisogno informativo: sono quattro...il bisogno informativo spesso evolve da uno stato basso(viscerale) che è uno stato vago, astratto.

- **Viscerale**: Esiste un need ma non viene espresso dall'utente; è tipicamente a livello interiore, uno stato d'animo, una forma di non-soddisfazione che non può essere espressa letteralmente.
- **Consapevole**: L'utente è mentalmente cosciente del proprio bisogno e riesce ad esprimerlo verbalmente anche se in modo poco chiaro e definito.
- **Formalizzato**: l'utente è in grado di dare una formalizzazione razionale del bisogno sebbene non sa se tale formalizzazione sia utile durante la ricerca di informazioni, che può essere fatta sia per mezzo di interazioni con altri individui che con un sistema di ricerca.
- **Compreso**: L'utente riesce a formalizzare il proprio bisogno con una richiesta formale (question) e con un linguaggio adatto ad essere interpretato da un bibliotecario o un sistema di ricerca (information system)

Ci interessano di più gli ultimi due (all'interno dell'IS Information System)

Formalizzato: non può ancora interagire con gli altri o formulare una query su un motore di ricerca che può fare solo con l'ultimo stato (che è quello che ci interessa di più).

Spesso uno degli obiettivi degli information systems è riconoscere i bisogni informativi degli utenti allo scopo di soddisfarli. Tale attività è più semplice se il bisogno è ad uno stadio

formalizzato o *compreso*; in tutte e due i casi infatti possiamo interagire con l'utente ed aspettarci di ottenere suggerimenti utili.

Nei livelli *consapevole* o *formalizzato* è sempre possibile interagire con l'utente ma le indicazioni fornite sono rappresentazioni imperfetti dei bisogni correnti. Occorre considerare un inevitabile grado di incertezza sulla rappresentazione dei bisogni che possiamo dedurre. In questi due stadi è importante utilizzare tecniche implicite, dove le azioni dell'utente vengono spiate e monitorate allo scopo di individuare comportamenti che possono dare indicazioni sui bisogni correnti.

Altra grossa caratterizzazione che ci aiuta a capire alcune classi di bisogno informativo: a breve o lungo termine:

breve termine: es. meteo per domani...estemporaneo...una volta acquisita l'info il bisogno informativo scompare...una volta ottenuta la info si risolve...finestra temporale breve. es. decidere come vestirsi.

lungo termine: lunghezza temporale più ampia, si ripetono, poichè essendo un bisogno informativo più ampio mi tocca interagire più volte con l'informazione. E' un task a lungo termine..bisogna interagire più volte con l'informazione.

Possibile caratterizzazione in ambito web dell'information needs.

Ogni volta che sottomettiamo una query ad un motore di ricerca dobbiamo capire cosa stiamo cercando qual è l'obiettivo.

- Obiettivo Navigazionale (voglio un indirizzo, una risorsa)
- Materiale informativo (un po' più importante), mi serve del materiale.
 - a) diretto--> apprendere qualcosa su un certo argomento
 - a1) chiuso (risposta ad una domanda con risposta univoca e non ambigua)
 - a2) aperto (mi approfondisce una domanda, la risposta non può essere diretta)
 - b) suggerimento: voglio un'idea, un suggerimento su un argomento
 - c) locazione: dove posso ottenere un servizio o un prodotto
 - d) lista: ottenere una lista di siti che possono aiutarmi per uno specifico goal
- Risorse: Download; divertimento; interazione...

Come possiamo caratterizzare i bisogni informativi : breve lungo termine in che modo possiamo ottenerli: Risorse, materiale informativo, obiettivo navigazionale...

Information Seeking

Attività che l'individuo intraprende per cercare informazioni.

Il processo di *Information Seeking* ha luogo quando si riconosce un *gap* nella conoscenza corrente che può motivare una persona ad acquisirne di nuova.

Esiste tipicamente un *obiettivo* da soddisfare:

- Apprendere conoscenza
- Eseguire un certo task

Piu' in generale e' una attivita' motivata di acquisizione di conoscenza da sorgenti informative selezionate opportunamente dall'individuo.

Information seeking parte dall'utente...stiamo in uno stato un po' più avanzato rispetto allo stato primordiale.

Quello che ci interessa di più è l'information seeking e il bisogno informativo.

Modello di information Seeking:

Il modello di Ellis si focalizza invece sulle relazioni tra *stage* (o *features*) che l'individuo intraprende durante la ricerca sebbene gli stessi autori notino come tali relazioni dipendano fortemente dal contesto specifico.

Il modello include 8 stages non necessariamente lineari temporalmente:

- **Starting**: consiste nelle attività preliminari effettuate prima di avviare una ricerca di informazioni, ad esempio identificare il mezzo iniziale utile per la ricerca. Attività preliminari, selezionare una sorgente piuttosto che un'altra, per caratterizzare l'utente che tipo di sorgente viene utilizzata, se già conosciamo diverse sorgenti possiamo già utilizzare una selezione, considerazioni per selezionare una sorgente rispetto ad un'altra...sarebbe opportuno modellarlo.

Entrano in gioco cose difficili da valutare:

La familiarità;

Autorevolezza

Qualità (interfaccia)

Accessibilità (se devo pagare per l'info...o se ci metto un'ora per ottenerla)

Quattro misure molto difficili da definirle chiaramente e da misurare.

Sono qualità individuali. Sono misure che dovrebbero essere molto contestuali riferite all'utente

(questi tre non hanno un ordine temporale: Chaining, Browsing, Monitoring)

-**Chaining**: Chaining attività molto importante, una delle poche che l'utente può fare all'interno del web.

due tipi di chaining:

(forward chaining)avanti:andare a vedere chi cita l'informazione/pagina/sito corrente, conosciamo questa informazione solo attraverso servizi esterni (base di dati in biblioteca.

(backward chaining)indietro:elemento della bibliografia, elementi informativi pubblicati prima

Informatico: avanti(forward chaining) perchè è interessato a cose più recenti, più affidabili rispetto al passato

Umanista: indietro, gli serve lo storico, le fonti, su cosa si basa...

-**Browsing**: seconda attività in parallelo:

Browsing: sembra l'attività vera e propria di seeking due tipologie:

across-document browsing: altri documenti

Within-document browsing: stesso documento lo stiamo facendo sullo stesso documento o su altri.

Stiamo ancora cercando informazione non stiamo apprendendo (quindi è browsing)

-Monitoring: è un po' più diretto. Continuiamo a dare un'occhiata alle info per selezionare le sorgenti più importanti...è un'attività che si ripete, ad es siti di congressi o riviste online che fanno uscire continuamente nuove info e noi monitoriamo periodicamente le nuove info(andiamo a vedere periodicamente se sono uscite nuove info).

-Differentiating: Differenziazione: a questo punto abbiamo collezionato l'informazione, ne abbiamo più di una(pagina web documento) dobbiamo fare quindi il filtraggio, quali sono le tipologie di interazione con l'informazione (le 7 fasi che abbiamo accennato)

-Extracting: identificare selettivamente materiale d'interesse da una sorgente informativa precedentemente individuata e approfondirlo.

-Verifying-Ending: controllare l'accuratezza dell'informazione e se sono presenti errori.

I *processi* di information seeking nel campo della computer science possono essere organizzati in base a due caratteristiche:

-Stabilità temporale del bisogno informativo

- *Stabile o dinamico*

-Specificità del bisogno informativo

- *Specifico*
- E.g., Qual è il candidato favorito nelle prossime elezioni?
- *Ampio respiro*
- E.g., Conoscere la vita di JFK

-Tipologia di dati prodotti dalla information source testuale

• *Strutturata* dove i dati sono conformi a un certo schema con una semantica chiara associata

ad ogni campo, e.g., record di una base di dati.

- *Non strutturata* dove l'informazione è tipicamente espressa in linguaggio naturale

Nel nostro contesto, per *processo* si intende una attività condotta da umani, eventualmente con l'assistenza di un sistema informatico.

Un *sistema* invece indica un sistema software automatizzato e relativa unità elaborativa sviluppata a supporto degli utenti durante un processo.

Perciò un sistema di Information Filtering è sviluppato per supportare gli utenti durante processi di Information Filtering.

Processo di Information Seeking suddiviso in due tipi: bisogno informativo e sorgente informativa.

Il bisogno informativo se è stabile ricadiamo in un processo di information filtering, se è dinamico facciamo difficoltà a metterlo in relazione l'uno con l'altro, (information Retrieval) I bisogni informativi dell'utente non sempre sono dinamici, non sempre abbiamo bisogno di soddisfare un nuovo bisogno informativo possiamo aver bisogno di colmare la stessa lacuna.

Information Filtering

In un processo di Information Filtering si assume che gli utenti mostrino bisogni informativi e interessi a *lungo termine* e le sorgenti contengano informazioni usufruibili direttamente dall'utente.

Tipicamente i sistemi software basati sull'Information Filtering devono gestire un *grosso volume* di informazioni, *generate dinamicamente* fornendo all'utente quelle che piu' verosimilmente soddisfano i suoi bisogni informativi.

Obiettivo: migliorare le capacita' dell'utente durante la ricerca di informazioni. Abbiamo un grosso volume di info generate dinamicamente...non ci aspettiamo che si ripetano spesso nel tempo (vd. le news). Vogliamo filtrare tra questa grossa mole le informazioni di interesse per l'utente.

La distinzione principale del processo di Information Filtering con quello dell'Information Retrieval (alla base degli attuali motori di ricerca Web) e' la velocita' di aggiornamento dei bisogni informativi e la velocita' di aggiornamento (e produzione) della sorgente informativa. Nel Filtering si suppone che il bisogno informativo sia piu' stabile mentre la sorgente produca informazioni con un tasso piu' rapido.

Un processo ideale dovrebbe poter affrontare sorgenti che si aggiornano spesso e bisogni informativi che possono variare in ogni istante.

Nel processo di Information Filtering ci sono tre fasi:

1. *Collection*: selezione a priori in cui non entra in gioco l'utente. Esistono due metodologie per collezionare le informazioni:
 - *Attiva*: esiste un modulo software che si occupa di ricercare l'informazione da analizzare
 - *Passiva*: l'informazione viene fornita per mezzo di uno stream, e.g., news feed
2. *Detection*: I software basati su IF tipicamente includono una componente di modellazione degli interessi dell'utente (*user model* o *profile*) il cui obiettivo e' individuare e rappresentare gli interessi a lungo termine dell'utente. E' una fase in cui entra in gioco l'utente e il bisogno informativo. (matching) argomento: bisogno informativo e documento ed una misura di similarita' (1: max attinenza; 0: minima attinenza) non solo i bisogni informativi ma anche altre informazioni, prevedere i bisogni informativi, su un ambito di ricerca che non si conosce...predire. Anche preferenze, doc. lungo o corto, la lingua.
3. *Display*: come visualizzare i documenti rappr. grafica o testuale, grafico a torta...nulla vieta all'inf system di analizz l'info e proporre un suo elaborato, ci sono gia' dei servizi on-line che costruiscono pagine dinamiche con l'informazione raggruppata. L'informazione data non e' quella originale, ma un suo processamento. Costruzione di un documento ex-novo a partire dalla query.
4. *Considerazione*: quando interrompere l'utente? Quando suggerire l'informazione, momenti in cui si e' interrompibili, altri no, dipende anche dal tipo di informazione (se di particolare interesse o no).

Information Retrieval

Il processo di IR e' fondamentale perche' e' alla base di molti sistemi di gestione dell'informazione come ad esempio i motori di ricerca, sebbene tale processo risulta incompatibile con il dominio.

Motivi: I motori di ricerca per il Web devono trattare una sorgente informativa per niente statica; La sequenza di query formulate da un utente non sempre sono scorrelate tra loro e variabili.

(architettura motore di ricerca)

Il motore di ricerca e' composto principalmente da quattro moduli:

1. Un *crawler* segue i link presenti nelle pagine Web ed effettua il download
2. La copia viene immagazzinata in un *repository* locale
3. Un *indexer* analizza il testo delle pagine e lo memorizza con una opportuna rappresentazione in una base di conoscenza; questo permette di massimizzare la velocita' nel retrieval
4. A partire da una query, un *query engine* costruisce una lista ordinata di risultati

Due soluzioni vengono normalmente impiegate per adattare il processo di Information Retrieval al contesto del Web.

1. Il motore di ricerca possiede una base di conoscenza dove memorizza con una opportuna rappresentazione copie delle pagine recuperate dal Web, in questo modo e' possibile velocizzare il processo di retrieval a partire da una query. Periodicamente si aggiorna questa base di conoscenza con le modifiche apportate sulle pagine Web.
 - **Attenzione:** la sorgente non e' più considerata statica.
 - **Svantaggi:** Tempo e risorse computazionali (network e storage) per mantenere aggiornata la base di conoscenza del motore di ricerca; Inoltre, a causa della rapidita' negli aggiornamenti del Web e dei limiti nella banda passante degli attuali network, la base di conoscenza sara' sempre non-sincronizzata rispetto all'informazione corrente sul Web.
2. L'insieme delle query formulate da un utente vengono considerate scorrelate l'una dall'altra
 - **Svantaggi:** Ignorare eventuali correlazioni tra le query utente non permette di analizzare eventuali evoluzioni o alterazioni dei bisogni informativi, percio' si ignorano aspetti importanti delle esigenze dell'utente.

Sistemi di Information Retrieval basati sul modello a spazio vettoriale sono vantaggiosi perche' la loro computazione puo' essere eseguita velocemente e produrre risultati in frazioni di secondo, ma mostrano alcuni grossi svantaggi a livello semantico:

Information Filtering VS Information Retrieval

IF e IR sono concettualmente dissimili:

- IF rimuove dati da uno stream dinamico di informazioni
 - Ad esempio: junk emails, news feeds filtering
- IR accede e recupera informazioni da collezioni statiche
 - Ad esempio: librerie digitali
 - Ma nella pratica spesso vengono riadattati per vari scopi,

La presenza di *user profiles* nel processo di IF implica *rappresentazioni più complesse* dei bisogni informativi rispetto al processo di IR.

Nel IR si studiano tecniche per *ridurre il tempo di recupero* mentre nel'IF si preferisce aumentare la precisione nei risultati utilizzando tecniche di matching tra bisogno informativo e informazione piu' sofisticate, che possono risolvere almeno in parte il problema dei false match.

Inizialmente i ricercatori hanno proposto molti approcci basati sul processo di IF, attualmente però si tende ad adattare sistemi di IR tradizionali incorporando modelli utente tipici del IF. I vantaggi di tale soluzione sono:

- Mantenere le attuali infrastrutture dei motori di ricerca ampiamente collaudate aumentando la precisione nei risultati includendo gli interessi dell'utente
- Si mantengono le stesse interfacce utente dei motori di ricerca, facili da comprendere per un utente
- Si sfruttano le funzionalità offerte dai moduli del motore di ricerca all'interno del processo di filtraggio, e.g.,:
 - Accedere alla storia delle query e dei risultati visionati dall'utente
 - Accedere alla collezione di pagine memorizzate all'interno della base di conoscenza
 - L'unico vincolo è riuscire a mantenere i tempi di risposta nella produzione dei risultati al di sotto di un limite prefissato (e.g., 0,5 secondi)

Esempio

Google Personalized tiene traccia di un profilo utente allo scopo di posizionare le pagine di maggiore interesse in cima.

Ad esempio, per un utente che ha visitato spesso articoli scientifici riguardo tecniche di IR e IF, si ottengono due liste di risultati distinti a partire dalla query "information retrieval".

Information Behaviour

Con il nome di *Information Behaviour* si indicano più in generale tutti i comportamenti dell'utente in relazione alle sorgenti e canali informativi, che includono anche i processi di information seeking attivi e passivi, e l'utilizzo dell'informazione stessa

Comportamento relativo all'informazione. È più generale dell'information seeking.

Qui troviamo anche i comportamenti passivi dell'utente con l'informazione più altri tipi di comportamenti che non riguardano l'information seeking...comunicazione face to face a volte non è rivolta all'acquisizione di conoscenza.

Non interessa andare oltre, interessa sapere che esistono altri tipi di interazione tra utente e informazione che noi dobbiamo considerare, ma che non possiamo monitorare né spiare, non riusciamo a includerle...dobbiamo considerare nel nostro modello utente dei buchi neri che noi non possiamo conoscere...l'utente interagisce con l'informazione anche fuori (information behaviour).

Con il termine *Information Searching Behaviour* si indicano l'insieme dei comportamenti micro-level che l'utente manifesta durante l'interazione con gli information systems volti alla ricerca.

Tipicamente comprende anche attività mentali quali esprimere giudizio sulla qualità e rilevanza delle informazioni recuperate

È una specializzazione del termine Information Behaviour.

Il termine *Information Use Behaviour* indica invece le azioni fisiche e mentali che vengono operate allo scopo di inglobare l'informazione all'interno della propria conoscenza

Modello Wilson Information behaviour, svantaggi: Non c'è alcun elemento per capire perché l'individuo si mette a cercare l'informazione...non c'è nulla che legghi l'information behaviour all'information need

Text Classification

Definizione Text Classification: L'assegnamento di un documento in una categoria presa da un insieme predefinito.

Inter-indexer inconsistency: Visto che la classificazione di un testo si basa sulla semantica, e dato che la semantica di un documento è una nozione soggettiva ne segue che l'appartenenza di un documento a una categoria non può essere decisa deterministicamente.

Vincoli al task della classificazione: per un dato k , esattamente k , ($0 \leq k, 0 \geq k$) elementi di C vengano assegnati ad un documento d_j

- Single Label: soltanto una categoria può essere assegnata a un documento ($k=1$)
- Multi Label: 0 o più categorie possono essere assegnate a un documento
- Binary: un documento o appartiene a ci o appartiene a $\neg ci$

Hard: il classificatore Φ_i restituisce un valore booleano

Ranking: il classificatore Φ_i restituisce un valore $[0,1]$

Indicizzazione automatica per sistemi di IR

Ad ogni documento è assegnata 1 o più parole-chiavi (descrittivi il suo contenuto) provenienti da un dizionario controllato. Solitamente questo lavoro era fatto a mano da indicizzatori umani.

Filtering di Testi

E' l'attività di selezionare una collezione dinamica (uno stream) di testi. Ad esempio una agenzia di stampa invia notizie agli utenti. Il sistema di filtering fa arrivare all'utente soltanto le notizie che gli interessano. E' un'applicazione che risale agli anni '60, ma l'esplosione della disponibilità di informazione digitale ne ha ingigantito l'importanza. Ad oggi è usato in moltissimi contesti: la creazione di giornali Web personalizzati, filtraggio di e-mail spazzatura ecc... Con Filtering adattativo si intende un filtering capace di adattarsi all'esigenze dell'utente che di volta in volta invia una valutazione del filtraggio

Approccio Knowledge Engineering

La creazione di un classificatore di testi automatico consiste nella creazione di un sistema esperto capace di prendere decisioni di classificazione.

Le regole venivano definite da un ingegnere della conoscenza con l'aiuto di un esperto del dominio

SVANTAGGI:

1. Se si deve modificare l'insieme di categorie è di nuovo necessario l'aiuto dell'esperto di dominio.
2. Se si vuole cambiare dominio del classificatore si deve chiamare un nuovo esperto di dominio e riniziare il lavoro da capo.

Approccio Machine Learning

Un processo induttivo costruisce automaticamente un classificatore per una categoria ci
Dalle caratteristiche osservate il processo induttivo decide quale caratteristiche deve avere un nuovo documento per essere classificato sotto ci

Non si costruisce un classificatore, ma un costruttore di classificatori che va bene per ogni dominio

RISORSA CHIAVE: Documenti classificati manualmente (spesso sono già disponibili ma anche se non sono disponibili...)

E' piu' facile classificare documenti manualmente piuttosto che stabilire delle regole per la classificazione dei documenti perché è spesso più facile caratterizzare un concetto estensionalmente piuttosto che intensionalmente

Il corpo iniziale dei documenti già classificati viene diviso in tre insiemi:

1. Training Set: Insieme dei documenti che vengono usati per costruire il classificatore
2. Validation Set: Una volta costruito il classificatore potrebbe essere necessario aggiustare dei parametri. Per valutare il giusto valore da assegnare ai parametri si fanno test su questo insieme
3. Test Set: Usato per testare l'efficacia del classificatore

I tre insiemi devono essere assolutamente disgiunti

La costruzione di un classificatore si articola in tre fasi:

1. Indicizzazione dei documenti e riduzione dimensionale
2. Induzione del classificatore
3. Valutazione dell'efficacia del classificatore

Indicizzazione dei documenti e riduzione dimensionale:

I documenti non possono essere interpretati direttamente da un classificatore.

Per questo si applica una procedura di indicizzazione che mappa un documento in una rappresentazione compatta del suo contenuto.

Un documento d_j viene rappresentato come un vettore di pesi

$$d_j = \langle w|j, \dots, w|T \rangle$$

T = insieme dei termini; $0 < w_{kj} < 1$ = quanto il termine t_k contribuisce alla semantica d_j

Si usano pesi binari o non binari (tf-idf)

Prima di indicizzare: eliminazione delle stop word, Stemming (raggruppare le parole per la loro radice morfologica)

Riduzione dimensionale: Si devono pertanto individuare i termini più interessanti per classificare. I termini più interessanti hanno frequenza medio bassa.

Per Selezione di termini:

- *Metodo di selezione dei termini wrapper:*

Si costruisce un classificatore per ogni T' (sottoinsieme dei termini T) possibile e si seleziona il T' che ha generato il classificatore più efficace. Metodo forza bruta: Costosissimo

- *Metodo di selezione dei termini filtering:*

Si prendono i $|T'|$ termini che hanno ricevuto il più alto punteggio da una funzione che misura l'importanza di un termine per la classificazione. Sono state definite e studiate molte funzioni per questo proposito.

Per Estrazione di termini:

Fissato un x , si cerca di sintetizzare dall'insieme dei termini T , un insieme T' (tale che $|T'|=x$) di nuovi termini che massimizzi l'efficacia. Uno dei metodi di estrazione dei termini usato per la classificazione di testi è il Term Clustering: si raggruppano le parole "simili" in cluster, in modo che i cluster possano essere usati (piuttosto che i termini) come dimensioni dello spazio vettoriale.

Induzione di classificatori:

Il problema della costruzione induttiva di un classificatore di testi è stato affrontato in svariati modi. Mostriamo i metodi più popolari.

La costruzione induttiva di un classificatore per una categoria ci consiste nel:

1. definire una funzione $CSV_i: \mathcal{D} \rightarrow [0,1]$ (CSV = Classification Status Value)
2. determinare una soglia τ_i

La funzione CSV_i prende un documento dj e restituisce un numero che rappresenta quanto dj dovrebbe essere classificato sotto ci

Se la classificazione è Hard allora il nostro classificatore è

$$\Phi_i = CSV_i > \tau_i$$

Se la classificazione è Ranking allora il nostro classificatore è

$$\Phi_i = CSV_i$$

Classificatori probabilistici vedono $CSV_i(dj)$ in termini di $P(ci|dj)$, cioè la probabilità che un documento, rappresentato da un vettore dj , appartenga alla categoria ci e cercano di calcolare tale probabilità utilizzando il teorema di Bayes:

$$P(ci|dj) = P(ci)P(dj|ci)/P(dj)$$

Dove lo spazio degli eventi è lo spazio dei documenti:

$P(dj)$ = probabilità che un documento preso a caso sia uguale dj

$P(ci)$ = probabilità che un documento preso a caso appartenga a ci

Classificatori Naive Bayesiani (uno dei più famosi: Binary Independence)

Il tempo per classificare un documento è lineare con il numero di termini

Classificatori simbolici

I classificatori probabilistici sono di natura essenzialmente numerica, e quindi spesso poco interpretabili direttamente da umani.

I classificatori simbolici non soffrono di questo problema. Tra questi troviamo:

1. Alberi di decisione

- I nodi interni sono etichettati con termini
- I rami che partono dal nodo interno x hanno un test sul peso del termine che etichetta x
- Le foglie sono etichettate da categorie

Un tale classificatore categorizza un documento d_j testando (secondo quanto indicato sui rami) ricorsivamente i pesi che i termini hanno nel vettore d_j , finché un nodo foglia non è raggiunto

Strategia “*divide and conquer*”

SVANTAGGI: Un albero “troppo allenato” può diventare troppo specifico (overfitting).

La

maggior parte dei metodi per l’induzione di Alberi di Decisione includono un metodo per far

crescere l’albero e un metodo per potarlo (per eliminare i rami troppo specifici)

2. Regole di Decisione

Un classificatore per una categoria ci è una regola in forma normale disgiuntiva (DNF).

Es:

IF[(\neg wheat) \wedge (bushels \vee \neg farm) \wedge (export)] THEN ci

Le Regole di Classificazione sono solitamente classificatori più compatti degli Alberi di

Classificazione.

Funzionano soltanto con documenti rappresentati come vettori di termini binari.

L’induzione di regole avviene in modo BOTTOM-UP: ogni documento del training set è visto

come una regola dove le clausole sono i termini del documento e la testa è la categoria in cui il

documento è inserito. Pertanto i documenti del training set formano un insieme di regole. Queste regole vengono poi generalizzate.

Metodi On-Line

Con la locuzione ‘algoritmo on-line’ si intende un algoritmo, per la risoluzione di un problema, che deve fornire dei risultati pur non avendo a disposizione, inizialmente, alcuni dei dati in ingresso.

Più precisamente, l’algoritmo riceve in ingresso una generica sequenza di input X di cui, per ogni termine x_i , deve fornire una risposta o prendere una decisione, basandosi solo sugli input ricevuti fino a tale istante, x_j con $j \leq i$.

A questa tipologia si contrappone la definizione di algoritmo off-line che designa i classici algoritmi, i quali possiedono tutti i dati d’ingresso fin dall’inizio dell’elaborazione ovvero tutta la sequenza X .

Vediamo nel seguito alcuni esempi di algoritmi on-line.

Expert Advice

Chiediamo ad n esperti di fornirci la previsione meteorologica di domani...

- Ogni esperto può esprimersi attraverso un valore nominale {rainy, sunny}
- Vogliamo analizzare il 'consiglio' degli esperti per ottenere la nostra previsione...

Algoritmo di Halving

Ipotizziamo di avere un esperto 'perfetto'

- Strategia: prendiamo il voto di maggioranza come previsione.

Ad ogni errore verificato, tagliamo gli esperti non affidabili.

- Al più $\log_2(n)$ errori:
 - caso peggiore: ogni errore taglia lo spazio degli esperti di un fattore 2.

Cosa succede se non esiste l'esperto perfetto?

- Strategia #1: iterare l'algoritmo di Halving.
 - Tutto come prima.. ma una volta che abbiamo tagliato via tutti gli esperti riapriamo dall'inizio.
 - Al massimo $\log_2(n) * OPT$ errori, dove OPT è il numero di errori del miglior esperto.
- Algoritmo poco furbo: perde periodicamente l'addestramento! ... si può fare di meglio!

Weighted Majority Algorithm

Intuizione: se l'esperto commette un errore lo penalizziamo (ma non lo escludiamo!).

- Anzichè tagliar fuori l'esperto, attribuiremo una penalità.
 - Start-up con esperti con peso pari a 1.
 - Predizione basata sulla scelta del voto di maggioranza (somma dei pesi).
 - Penalità per gli esperti che commettono un errore, moltiplicando per 0,5 il relativo peso.

Il numero di errori commessi dall'algoritmo WM è al più $2.41(m + \ln(n))$

$n = \#$ esperti

$M = \#$ errori commessi da WM

$m = \#$ errori commessi dal miglior esperto

$W =$ peso totale (parte da n) di tutti i pesi considerati per ogni esperto

$2.41(m + \ln(n))$

dopo M errori W decresce al più del 25%

perchè?--> il caso peggiore è che la metà dei classificatori sbaglia classificazione. siccome moltiplico per 1/2 quelli che sbagliano decremento della metà la metà dei classificatori.

Dopo M errori il peso sarà al più : $W \leq n * (3/4)^M$

il peso del miglior esperto è 1/2 (la sua penalità) alla m .. quindi vedere i calcoli sulle dispense...

ci da modo di valutare in modo empirico quant'è M

quanto sbaglia l'algoritmo ci da l'upperbound in funzione del miglior esperto.

Winnow Specialist

Ogni specialista parte con peso 1

- Predizione con WM voting
- Se l'algoritmo globale commette un errore,
 - Moltiplica lo specialista non corretto con 0,5
 - Moltiplica lo specialista corretto con 2

Classificatori Lineari

Un classificatore lineare per una categoria ci è un rappresentazione di ci in termini di un vettore $ci = \langle w1i, \dots, w|T|i \rangle$ nello spazio $|T|$ - dimensionale (lo stesso dei documenti).

$CSVi(dj)$ è il dot product tra dj e ci

I metodi per la costruzione induttiva di classificatori lineari sono divisi in due categorie:

1. On-line Method (ex: Perceptron)
2. Batch Method (ex: Rocchio)

Metodi On-Line per l'induzione di classificatori lineari

IDEA: Si definisce il classificatore dopo aver analizzato il primo training document e con i successivi documenti si raffina il classificatore

Molto utile:

1. se inizialmente non è disponibile interamente il training set
2. in quelle applicazioni in cui l'utente del classificatore provvede un feedback su come i documenti sono stati classificati (filtering adattativo).

Metodo di Rocchio per l'induzione di classificatori lineari

Il metodo di Rocchio è un metodo Batch (???), che tutto in una volta induce un classificatore

$$Ci = \langle w1i, \dots, w|T|i \rangle$$

I pesi vengono calcolati con la seguente formula:

$$wki = \alpha (\sum \{dj \in POSi\} wkj / |POSi|) - \beta (\sum \{dj \in NEGi\} wkj / |NEGi|)$$

α e β sono due parametri controllati che permettono di valutare l'importanza degli esempi negativi e degli esempi positivi

$$POSi = \{dj \in Tr \mid \Phi I(dj, ci) = T\}$$

$$NEGi = \{dj \in Tr \mid \Phi I(dj, ci) = F\}$$

In generale il classificatore Rocchio guarda la vicinanza del documento da classificare al centroide degli esempi positivi e la lontananza dal centroide degli esempi negativi.

SVANTAGGI: Il classificatore Rocchio come tutti i Classificatori Lineari ha lo svantaggio che divide lo spazio dei documenti linearmente. Questo comporta gravi perdite di efficacia: la media è solo parzialmente rappresentativa dell'intero insieme.

Algoritmo di Rocchio (apprendimento)

Sia l'insieme delle categorie $\{c1, c2, \dots, cn\}$

For i from 1 to n let $pi = \langle 0, 0, \dots, 0 \rangle$

(inizializzazione)

For each esempio di training $\langle x, c(x) \rangle \in D$

Let $d =$ vettore TF/IDF per il doc x

Let $i = j: (cj = c(x))$

(somma di tutti i vettori in ci per ottenere pi)

Let $pi = pi + d$

Algoritmo di Rocchio (test)

Dato un documento di test x
 Let $d =$ vettore TF/IDF per x
 Let $m = -2$ (inizializzazione)
 For i from 1 to n :
 (calcola la similarità col vettore prototipo)
 Let $s = \text{cosSim}(d, p_i)$
 if $s > m$
 let $m = s$
 let $r = c_i$ (aggiorna il più simile)
 Return class r

Classificatori basati su esempi

IDEA: Non si costruisce una rappresentazione della categoria, ma si confida sui documenti del training set che sono più vicini al documento che vogliamo classificare.

K-NN(K nearest neighbours)

IDEA: Per decidere se classificare d_j sotto c_i k -NN guarda se i k documenti del training set più simili a d_j sono stati classificati sotto c_i . Se una parte grande abbastanza è stata classificata sotto c_i allora il documento viene classificato in c_i altrimenti no .

La similarità tra documenti è calcolata con una qualche funzione $RSV(d_i, d_z)$.
 Nell'implementazione di Yang si cerca di dare più peso ai documenti più vicini.
 $CSV_i(d_j) = \sum_{d_z \in \text{Trk}(d_j)} RSV(d_i, d_z) [\Phi I(d_z, c_i)]$
 $\text{Trk}(d_j) = i$ k documenti del training set più simili a d_j secondo la funzione RSV
 $[x] = 1$ se x è TRUE, 0 se x è FALSE

La soglia k viene determinata empiricamente attraverso test sul validation set. E' stato dimostrato che aumentando di molto k non diminuiscono significativamente le performance

VANTAGGI: k -NN, diversamente dai classificatori lineari non suddivide lo spazio dei documenti linearmente. Quindi risulta essere piu' "locale".

SVANTAGGI: Inefficienza a tempo di classificazione: k -NN deve calcolare la similarità di tutti i documenti del training set con il documento da classificare.

E' conveniente utilizzarlo per document-pivoted categorization: calcolare la somiglianza dei training document può essere fatto una volta per tutte le categorie.

Algoritmo di apprendimento Nearest-Neighbour

L'apprendimento si riduce al modo di immagazzinare le rappresentazioni degli esempi di training in D .

- Test dell'istanza x :
 - Elabora la similarità tra x e tutti gli esempi in D .
 - Assegna ad x la categoria del più simile in D .

- Non si calcolano esplicitamente i prototipi delle categorie.
- Conosciuto anche sotto il nome di:
 - Case-based
 - Memory-based
 - Lazy learning

Nearest neighbor si basa su **una metrica di similarità** (o distanza)

- La più semplice per uno spazio continuo è la distanza euclidea.
- La più semplice per spazi d'istanza m-dimensionali binari è la distanza di Hamming
- Per i testi, la similarità basata sul coseno, per i vettori costruiti mediante indicizzazione TF-IDF, è tipicamente la più efficiente.

Training:

For each each esempio di training $\langle x, c(x) \rangle \in D$

Calcola il corrispondente vettore TF-IDF, dx , per il doc x

Test dell'istanza y :

Calcola il vettore TF-IDF d per il doc y

For each $\langle x, c(x) \rangle \in D$

Let $sx = \text{cosSim}(d, dx)$

Ordina gli esempi, x , in D al decrescere di sx

Let $N =$ I primi k esempi in D .

(*ottiene così i vicini più simili*)

Return la classe con più esempi in N

Support Vector Machines (si veda dispensa dopo)

Non riesco a dividere iperpiano con lo xor, quindi aggiungo una dimensione fittizia, così divido in due ipersolidi e trovo la divisione perfetta.

Massimizzano il margine attorno all'iperpiano di separazione.

La funzione di decisione è completamente specificata da un sottoinsieme di esempi di apprendimento, i vettori di supporto.

SVMs si riconduce a risolvere un problema di programmazione quadratica.

Il metodo più performante per la text classification.

Valutare un classificatore di testi

La valutazione dei classificatori di testi è effettuata sperimentalmente piuttosto che analiticamente. La Classificazione di Testi non può essere formalizzata (a causa della sua natura soggettiva) e quindi non può essere valutata analiticamente. La valutazione sperimentale di un classificatore solitamente misura la sua efficacia: l'abilità di prendere la giusta decisione di classificazione.

Misure dell'efficacia di classificazione di testi:

π_i [Precision wrt ci] = $P(\Phi I(dx, ci) = T | \Phi(dx, ci) = T)$

indica il grado di Correttezza del classificatore rispetto alla categoria ci

ρ_i [Recall wrt ci] = $P(\Phi(dx, ci) = T | \Phi I(dx, ci) = T)$

indica il grado di Completezza del classificatore rispetto alla categoria ci

Le probabilita' vengono stimate sui risultati del classificatore su un test set
 $\pi_i = TP_i / (TP_i + FP_i)$ $\rho_i = TP_i / (TP_i + FN_i)$

TP_i = Veri Positivi= #documenti classificati correttamente sotto c_i

FP_i = Falsi Positivi= #documenti classificati incorrettamente sotto c_i

VN_i = Veri Negativi= #documenti non classificati correttamente sotto c_i

FN_i = Falsi Negativi= #documenti non classificati incorrettamente sotto c_i

Category c_i		expert judgments	
		YES	NO
classifier judgments	YES	TP_i	FP_i
	NO	FN_i	TN_i

π_i e ρ_i sono misure di efficacia relative alla categoria c_i .

Vogliamo definire l'efficacia di un Classificatore Globalmente.

π global precision e ρ global recall

Le misure π e ρ prese singolarmente non bastano per esprimere l'efficacia:

Il classificatore che classifica tutti i documenti sotto c_i ha $\rho = 1$ (non ci sono falsi negativi)

Il classificatore che classifica tutti i documenti sotto $\neg c_i$ ha $\pi = 1$ (non ci sono falsi positivi)

π e ρ sono inversamente proporzionali, per valutare l'efficacia di un classificatore si deve trovare la giusta combinazione di queste due misure: anche per questo scopo sono state elaborate

numerose funzioni di combinazione.

Sono state proposte misure diverse dall'efficacia per valutare un classificatore: efficienza, utilità (che tiene conto dei concetti di guadagno e perdita) ecc...

Qual è il miglior classificatore?

La risposta è relativa all'applicazione in cui il classificatore deve essere usato

Esistono collezioni di documenti standard su cui effettuare il BENCHMARK di un classificatore

Ma per selezionare il classificatore da usare non possiamo utilizzare i risultati dei benchmark ottenuti in letteratura.

Web Document Modeling LSI SVM

I metodi di ranking tradizionali calcolano l'attinenza di un documento ad una query sulla base della presenza o meno di parole contenute nella query: un termine o è presente o non lo è.

Latent Semantic Indexing

Nel **Latent Semantic Indexing** (LSI) la ricerca avviene per concetti: ma un concetto non è l'astrazione-generalizzazione di un termine (es: *golf*, *vestiario*) bensì un insieme di termini correlati (golf, maglia, vestito) detti co-occorrenze o **dominio semantico**.

Data una collezione di documenti, LSI è in grado di rilevare che alcune n-uple di termini co-occorrono frequentemente (es: gerarchia, ordinamento, classificazione).

Se viene fatta una ricerca con gerarchia/ordinamento, vengono automaticamente recuperati documenti che contengono anche (ed eventualmente solo!) classificazione (ESEMPIO SU DISPENSE GOLF, CAR, TOPGEAR, PETROL)

LSI è un metodo di recupero e indicizzazione che usa una tecnica matematica chiamata SVD (singular Value Decomposition) per identificare pattern nelle relazioni tra i termini ed i concetti contenuti in una collezione di testi non strutturata. LSI è basata sul principio che parole usate nello stesso contesto tendono ad avere simile significato. Un punto di forza di LSI è la sua abilità nell'estrarre il contenuto concettuale del testo stabilendo associazioni tra quei termini che occorrono in contesti simili.

Matrici delle co-occorrenze

Se L è una matrice $n \times m$ (termini \times documenti)

Allora:

$L^T L$ (L^T è la trasposta) è la matrice le cui righe ai rappresentano **le co-occorrenze di termini fra di e d_j** , per ogni d_j . Dato un documento, indica quali sono i documenti più simili.

[E' una matrice $m \times m$ (documento \times documento) e dato un documento dice quali sono i documenti più simili.]

$L L^T$ (L^T è la trasposta) è la matrice le cui righe ai rappresentano **le co-occorrenze nei documenti fra t_i e t_j** per ogni t_j . Dato un termine, indica quali sono i termini più correlati.

[Duale: matrice $n \times n$ (termini \times termini) nella quale vengono evidenziate le concorrenze dei termini (per un certo termine verifico quali altri termini co-occorrono all'interno di un documento)]

Usando, ad esempio, la matrice $L L^T$ potrei “*espandere*” ogni termine con quelli aventi il più alto valore di correlazione (cioè, aggiungere alla query in cui compare la parola w anche quelle che co-occorrono con w più frequentemente).

Come detto, LSI proietta la matrice L termini-documenti in uno spazio concettuale di dimensioni ridotte, dove le dimensioni sono gruppi di concetti che co-occorrono, definendo un “dominio semantico”.

Il metodo utilizzato per effettuare questa proiezione è la singular value decomposition, un metodo algebrico.

LSI non viene applicato spesso perchè bisogna risolvere dei problemi troppo complicati.

Se il dominio è ristretto e non troppo variabile queste metodologie possono essere applicate.

SVD

Abbiamo una matrice rettangolare.

Dobbiamo diagonalizzare una matrice rettangolare

prendiamo tre matrici che moltiplicate tra loro mi danno la matrice originale (decomponiamo la matrice originale in tre matrici).

La matrice s è quadrata. la matrice $k \times k$ azzerò gli autovalori, azzerò la matrice dal basso verso l'alto quindi inutile popolare la parte grigia delle altre matrici e rimoltiplico ---> in un certo senso ho ridotto lo spazio (è quello che succede bene o male nella codifica bitmap). Ho cmq una perdita ridotta di informazione

$$t \times d = (t \times k) * (k \times k) * (k \times d)$$

$$X' \quad T \quad S \quad D'$$

X' non è quindi proprio identica a x ...ma con una leggera perdita di informazione.

Faccio una cosa simile a quella per le immagini però uso un vettore termine documento...faccio la media rispetto a quello che succede intorno ai termini.

Creiamo i collegamenti tra i termini che co-occorrono e li togliamo a quelli che non co-occorrono.

Scopo SVD:

Trovare relazioni latenti nel grafo

•Esaltare i CONCETTI:

Scorpendo termini polisemici

Accorpendo termini sinonimi

SVD: voglio spostarmi in uno spazio dei concetti, lo faccio tramite trasformazioni lineari. Devo spostarmi in un punto in cui riesco a valutare le co-occorrenze.

SVD in LSI in conclusioni

Nel modello vettoriale, queries e documenti sono vettori in uno spazio le cui dimensioni sono i termini, considerati fra loro ortonormali, cioè indipendenti fra loro

LSI trasferisce questi vettori in uno spazio le cui dimensioni sono concetti, cioè co-occorrenze fra termini

La riduzione di rango ha l'effetto di eliminare i concetti poco rilevanti.

Similarità query documento

Tecnica basata su decomposizione matriciale che permette di ridurre lo spazio in base a quanto scelgo k (se riduco troppo lo spazio perdo troppa informazione) l'obiettivo è trovare il k ideale riduce lo spazio e lavora cmq sui concetti. Lo svantaggio è che questa roba presuppone che uno risolva un'equazione dell'ordine della grandezza dello spazio se lo spazio è $m \times m$ l'equazione è di ordine m ---> tempo molto lungo. Quindi o ho un dominio piccolo o grande e fissato (cioè che non cambia).

Classificatori a supporto vettoriale Support Vector Machines (SVMs)

Mi permette di trovare un iperpiano che divida le istanze positive da quelle negative.

Il problema è che se sto in uno spazio non linearmente separabile devo trovare un meccanismo che attraverso una trasformata mi porta in uno spazio linearmente separabile (aggiunge una dimensione mi pare).

Non solo trovare l'iperpiano, ma trovare l'iperpiano ottimo, dobbiamo trovare un iperpiano ottimo: che massimizzi lo spazio che c'è tra l'iperpiano e il primo esempio negativo/positivo.

Problemi **multilable**: le istanze negative sono molte di più di quelle positive.

Prerequisiti di SVM trovare due spazi equilibrati e bilanciati.

Si è dimostrato che la capacità di generalizzazione di questo classificatore cresce al crescere del margine (spazio tra l'iperpiano ed il primo esempio negativo/positivo).

Di conseguenza, la capacità di generalizzazione massima è quindi data dall'iperpiano a margine massimo, detto Optimal Separating Hyperplane (OSH).

Per come è stato definito, il classificatore a supporto vettoriale non può gestire casi in cui le classi non siano linearmente separabili. Per risolvere questo problema ci sono due approcci:

- rilassare i vincoli di corretta classificazione.
- considerare altre superfici oltre l'iperpiano.

Generalizzazione dell'iperpiano ottimo:

Nella fase di ottimizzazione si rilassa il vincolo di esatta classificazione dei campioni del training set (soft margins), introducendo delle variabili di slack.

Le variabili di slack dicono che potrebbero esserci degli esempi negativi che vanno dall'altra parte... voglio trovare il miglior iperpiano che posso questo errore è definito dalle variabili di slack che sono variabili di tolleranza, voglio quindi dare una tolleranza all'errore.

entra a far parte della modellazione il parametro C. Se C è infinito l'iperpiano diventa perfettamente separabile... se C è basso tollerero' un sacco di errori.

Casi non linearmente separabili

Nel caso in cui non sia la soluzione (insiemi linearmente non separabili), si introduce un mapping ad uno spazio di dimensione molto più grande in cui gli insiemi corrispondenti siano linearmente separabili.

Quindi, invece di aumentare la complessità del classificatore (che resta un iperpiano) si aumenta la dimensione dello spazio delle features.

In dipendenza della dimensione dello spazio in cui è formulato il problema originale, il mapping può portare anche a dimensioni molto elevate (circa 10^6) dello spazio trasformato.

Definizione delle caratteristiche di una SVM

Quindi se mi trovo in uno spazio non linearmente separabile non posso usare le variabili di slack utilizzo un kernel mi sposto in uno spazio linearmente separabile tramite una trasformazione.

Per impiegare una SVM è allora necessario definire:

- tipo di kernel da impiegare
- parametri del particolare kernel
- valore di C

Non esistono criteri teorici per queste scelte; tipicamente va fatta una verifica su un insieme di validazione.

Introduzione allo User Modeling

La modellazione dell'utente e' l'ambito di ricerca che ha lo scopo di riconoscere e rappresentare elementi caratteristici dell'utente o simulare processi cognitivi utili per migliorare l'interazione uomo-computer.

Per mezzo della profilazione (profiling) vengono istanziati user profiles associati ad un utente impiegati nei software.

Nell'ambito dei SII un modello di un utente puo' rappresentare informazioni relative ad una o piu' delle seguenti categorie:

- Conoscenza
- Interessi
- Goals e Tasks
- Background
- Caratteristiche peculiari
- Contesto di lavoro

E' possibile caratterizzare un modello utente in base a varie caratteristiche.

Un modello utente puo' essere:

- **Stereotipato o feature-based(individuali)**: se ogni modello fa riferimento a gruppi di utenti con caratteristiche simili tra loro oppure a singoli individui. Modelli stereotipati vengono anche utilizzati per l'inizializzazione dei profili utente (come discusso piu' avanti). I modelli utente **feature-based (o individuali)** rappresentano caratteristiche peculiari dell'utente, quali la conoscenza, gli interessi, i goal. Durante l'interazione col sistema tali features possono variare e occorre tenere aggiornata la relativa rappresentazione.

I modelli basati su **stereotipi** puntano a raggruppare utenti simili in cluster.

- Esiste una funzione di mapping che ha come parametro le features dell'utente e restituisce il cluster piu' affine.

Gli utenti all'interno di un certo cluster vengono trattati allo stesso modo da parte del sistema informativo. Percio' l'utente viene rappresentato unicamente dal cluster in cui e' contenuto.

- C'e' una approssimazione che tratta due utenti ugualmente anche se non hanno le medesime features.

- **Dato o inferito**: se il modello viene costruito in base alle indicazioni di un utente (ad esempio per mezzo di tecniche di explicit user feedback) o amministratore esterno oppure viene

costruito automaticamente in base ad una analisi dei dati relativi alle interazioni utente-computer (implicit feedback).

- **Implicito o esplicito**: se la modellazione ha luogo in vari punti di un sistema software senza la progettazione di una componente di modellazione ad hoc oppure se esiste tale componente indipendente (attenzione a non confonderlo con le tipologie di feedback).

- **Statico o dinamico**: se il modello rimane stabile per tutta l'esecuzione del sistema oppure se si adatta a variazioni nell'ambiente o nelle caratteristiche dell'individuo.

- **Breve termine o lungo termine**: se l'istanza del modello perdura limitatamente alle sessioni di lavoro corrente o se rimane valida per più sessioni, anche non consecutive tra loro.

- **Adattabile o adattivo**: sistemi con modello dato, cioè costruito esplicitamente dall'utente, che fa perlopiù riferimento a caratteristiche dirette dell'interfaccia, come la lingua o la scelta delle componenti da utilizzare: in tale caso si parla di *adattabilità* o *customizzazione* dell'interazione. Nei modelli *adattivi* (o adattativi) il modello viene inferito automaticamente dal sistema che fornisce una *personalizzazione* autonoma o parzialmente autonoma (i.e., con la supervisione dell'utente) dell'interazione.

Customizzazione: la scelta degli elementi informativi da visualizzare avviene esplicitamente da parte dell'utente.

Personalizzazione: le informazioni suggerite dipendono dal modello utente costruito automaticamente in base alle azioni fatte dall'utente.

Modellare la conoscenza

Modello scalare

La conoscenza dell'individuo è uno degli elementi più rilevanti nella fase di modellazione. L'interazione con l'informazione infatti è tipicamente motivata da una mancanza di conoscenza e il suo processamento porta ad una variazione della conoscenza stessa.

La conoscenza può alterarsi incrementalmente ma può anche ridursi col tempo (*forgetting*). Il modo più semplice per rappresentarla è per mezzo di un valore in una scala predefinita:

- *Quantitativa*, E.g., 1, 2, ..., 5
- *Qualitativa*, E.g., good, average, poor, none

Tipicamente tali valori sono prodotti da una fase di:

- *Auto-valutazione*, dove l'utente esplicitamente segnala il livello di conoscenza
- *Valutazione oggettiva*, quando invece esiste un esame formale (quiz, etc) della conoscenza

Sfruttando tali modelli utente è possibile riconoscere la tipologia dell'utente (per mezzo di stereotipi) ed associarlo ad una categoria, e.g, esperto, intermedio, novizio, e proporre ad esempio diverse versioni di una pagina Web, o suggerimenti aggiuntivi a seconda del livello di conoscenza.

Gli svantaggi del modello utente basato su valori in scala sono la scarsa precisione della rappresentazione, e.g.:

- Un valore di buona conoscenza sul dominio del *word processing* può corrispondere ad un valore basso sul sotto-dominio *formatting*.

Tipicamente i valori sulla conoscenza in un dominio sono una media, perciò valori da considerarsi approssimati.

Modello strutturale

Nel modello strutturale il dominio e' suddiviso in frammenti indipendenti che vengono valutati singolarmente.

Ogni frammento contiene due tipi di informazione:

1. Uno dei due tipi di conoscenza che e' rappresentata dal frammento:
 - *Procedurale* (problem-solving skills): conoscenza che puo' essere direttamente impiegata per lo svolgimento di un certo task
 - *Dichiarativa* o *Descrittiva* (fatti e relazioni tra di essi): conoscenza che puo' venire rappresentata da frasi dichiarative o proposizioni, e.g., logica primo ordine o semantic networks.
- 1.
2. Il confronto del livello di conoscenza con quello relativo ad un utente esperto rappresentato da un modello chiamato: *domain model* o *expert model* o modello *ideal student*.

Il piu' popolare modello strutturale e' l'*overlay model*.

- Rappresenta la conoscenza dell'utente con un sottoinsieme pesato della conoscenza di un utente giudicato esperto.

L'Overlay model e' ampiamente utilizzato negli Intelligent Tutoring Systems (ITS):

L'Overlay model mostra limiti nel rappresentare correttamente la conoscenza:

- L'utente puo' apprendere con un processo di "filling the gaps", incrementando la propria conoscenza, ma in altre circostanze l'utente puo' dimostrare di risolvere problemi sfruttando processi di generalizzazione e raffinamento a partire dal conoscenza esistente.
- A volte si impiega un *bug model* per rappresentare oltre alla conoscenza esatta anche quella errata: c'e' differenza tra non conoscere e conoscere in modo errato.

Weighted Vector of keywords

Una delle piu' popolari tecniche per modellare gli interessi e' per mezzo di un vettore pesato di parole chiave:

- Si associa un valore scalare ad insieme di parole corrispondente alla misura dell'interesse mostrato dall'utente per il concetto associato alla parola stessa.

Lo svantaggio principale del vettore delle keyword e' che soffre del ***problema del vocabolario***:

- Le persone raramente utilizzano gli stessi termini per indicare lo stesso concetto:
 - Un termine incluso in una pagina Web da un certo autore potrebbe non coincidere con la keyword espressa dall'utente con lo scopo di recuperarlo. Si dice anche che esiste *un mismatch tra il query space e il document space*.

Utilizzando il linguaggio naturale per rappresentare un certo dominio occorre considerare la presenza di:

- *Sinonimia*: due termini distinti che indicano lo stesso significato

- Computer, Elaboratore, Personal Computer, etc.
- *Polisemia*: un termine con significati distinti, e.g.,
 - Leopard: animale o sistema operativo?
- Termini con significati piu' generici (ipernomia) o piu' specifici (iponimia)
 - Veicolo, Autovettura, Maserati, Maserati Ghibli, etc.

Possiamo quindi costruire un modello utente usando un Vettore di termini pesato che rappresenta gli interessi dell'utente. Il problema sta nel modellare gli interessi con delle keyword

(DA IMPARARE)

Gli effetti del problema del vocabolario sono molteplici:

Primo caso: presenza di sinonimi: mettiamo nel vettore un solo termine che mi rappresenta il concetto.

Il problema è che tutti i documenti in cui l'autore ha usato un altro vocabolario non vengono recuperati dal motore di ricerca..quindi ogni volta che abbiamo dei sinonimi e non li inseriamo nel vettore che ci interessa abbiamo perdita di recall: [misura che tendo a massimizzare (tutti i risultati sono attinenti)].

Secondo caso: se nel vettore inserisco termini polisemici, nella mia lista di risultati ci saranno documenti che ricopriranno diversi concetti, non quello iniziale, ma anche tutti gli altri. Tra i miei risultati avro' anche documenti che non mi interessano(perdita di precision).

Terzo caso: Se sottomettiamo una query non sottometeremo mai una query con tutti i possibili termini che sono più specifici di quello immesso, il motore di ricerca ignorerà i documenti che contengono termini più specifici, non essendoci un match esatto esso non li includerà nei risultati--> avro' una perdita di precisione(perdita di precision).

Concept-level models

Di solito quindi si passa ad un livello superiore, si tende a far salire da un livello completamente sintattico ad un livello semantico. Sostituiamo il nostro mondo fatto di parole chiave con un mondo fatto di concetti. Al posto di un vettore di keyword ho un vettore di concetti

Il problema di fondo è che per rappresentare il mondo a concetti qualcuno deve indicare questi concetti (i documenti sono scritti con parole chiave non concetti...serve un mapping). Per questo il semantic web non riesce a decollare.

Questo mapping non è facile. Questi modelli basati su concetti sono molto adatti in domini chiusi in cui sostanzialmente il dizionario dei concetti/dei termini è un dominio limitato che non si espande e che non si apre.

Nei domini web ogni secondo c'è un concetto che viene inserito. E' un mondo nettamente variabile, in cui i concetti vengono spesso aggiornati e variati.

Quando abbiamo molti documenti utilizziamo ancora una volta un vettore basato su keyword prendiamo tutti i documenti che abbiamo. Se andiamo a prendere il dizionario dei motori di ricerca possiamo creare un vettore di una dimensione molto elevata a costo zero...sono info che abbiamo già lo estraiamo e creiamo il nostro modello di interesse basato sulle keyword. (quindi si usa il vettore di keyword).

Modellare Goal e tasks

Sono attività in cui abbiamo immediatezza..sono qualcosa da realizzare in maniera immediata e che una volta realizzato non si ripete nel tempo. Un bisogno informativo immediato potrebbe essere un goal.

Per riconoscere i goal correnti spesso si utilizza l'approccio *goal catalog*. (E' una tassonomia...cerchiamo di spezzettare un goal in varie parti indipendenti, serve anche qui un esperto del dominio). Lo si fa soprattutto in interfacce grafiche per determinati scopi (tipo ambito militare)..concept di un aereo, sono interazioni in domini molto più limitati.

Quello che si fa è una correlazione del task model che non rappresenta l'utente, ma il task.

E. organizziamo un viaggio, tendo a spezzettarlo in sottobiettivi abbastanza indipendenti.

Se il sistema intelligente potesse ottenere questa informazione potrebbe sfruttarla in molti contesti, sarebbe molto utile, se sapessi cosa sta cercando di fare l'utente potrebbe intervenire nelle sottoattività per supportare l'utente.

Percio' in questo caso modellare l'utente significa riconoscere il particolare goal/attività dell'utente.

Goal/task recognition

E' un processo complesso per le seguenti problematiche:

- Complessita' dei goal
- Human-computer interaction non sufficientemente informativa

Il goal non è mai uno ce ne stanno sempre una moltitudine, è difficile...dobbiamo quindi osservare l'utente nelle sue attività. Dobbiamo associare attività-goal. Abbiamo goal multipli. Problema delle influenze...a volte una certa azione si ripercuote su più goal contemporaneamente. Posso associare una attività a due goal differenti.

In alcuni casi le interazioni uomo-computer non sono sufficienti per riconoscere il goal corrente, per due motivazioni:

1. L'interfaccia non possiede molti metodi per interagire:
 - a.Ad esempio, nel caso del browsing Web ad esempio abbiamo un numero ridotto di possibilità di interazione rispetto ad un normale strumento software lato desktop.
 - b.Altri software possiedono interfacce più complesse, con molte più azioni disponibili e relative semantiche associate.
2. Le azioni dell'utente non hanno una semantica chiara:
 - a.nel dominio Web la scelta di un link in una pagina dipende da cosa l'utente si aspetta di trovare visitando la pagina destinazione. Gli elementi che ha a disposizione l'utente per determinare il contenuto della pagina destinazione sono: il testo del link ed eventualmente la struttura che lo contiene.

Tassonomia dei link:

Aggiungere a href un elemento category, semantica associata al link...perchè l'autore ha messo quel link. Abbiamo due tipologie di link:

- **Organizzativi:** generano la struttura "superficiale" (i.e. la posizione delle varie unità informative) del documento (pagina intero sito Web) e permettono all'utente la relativa

navigazione. E' il tipo più comune perchè siamo in ambito web. Il fatto di interagire con interfacce e fare formattazione non è un goal è un goal relativo con l'interfaccia. Minimizzare lo sforzo produttivo per fruire l'informazione. perciò' ogni volta che parliamo di un task model per interagire con l'interfaccia vuol dire che abbiamo fallito, l'utente non dovrebbe sprecare tempo per navigare la pagina (l'interfaccia non ha strumenti efficaci per fruire l'informazione).

- **Content based**: si riferiscono al significato del testo; esistono tre tipologie: semantic, rethorical e pragmatic links.

1- trattano il significato e le relazioni tra parole o, più' in generale, tra frasi, paragrafi, sezioni etc.

Relazioni semantiche notevoli sono: *similar, contrast, part/kind of*.

2- Rethorical: usato per accompagnare il lettore tra gli elementi informativi tipicamente con lo

scopo di fare apprendere un certo learning goal, supportare una certa tesi o opinione.

Mette

in relazione *semanticamente* informazioni.

Normal: Uno dei sottocasi notevoli dei link retorici quando l'autore vuole supportare una tesi, (un goal sottostante).

Commentary: Lato utente: lasciare un commento su un certo documento. Possiamo in questo caso sfruttare qualcosa di più per raggiungere un modello di goal? Potremmo inferire qualcosa di più sui suoi goal quando un utente clicca su un certo link.

3- Pragmatic: L'autore inserisce il link per suggerire del materiale.

Modellare il Background

Con background si indica *un insieme di caratteristiche* correlate all'esperienza pregressa dell'utente escludendo il *core domain* del sistema Web. Con background si intende che ci ficco dentro tutto quello che non è il dominio di interesse.

Nel background info relative all'utente ma non direttamente relazionabili all'obiettivo finale.

Professione; Qualifica; esperienze lavorative in aree correlate.

I modelli utente basati sul background sono simili a quelli basati sulla conoscenza, sebbene la rappresentazione utilizzata per i background e il relativo impiego sono diversi.

- Poichè' informazioni dettagliate sul background sono inutili, solitamente si abbandona il modello overlay per un più' semplice schema a stereotipi.

- Inoltre il background ci considera stabile durante l'interazione col sistema ed è difficile dedurlo unicamente guardando le azioni dell'utente. Per tale motivo esso viene comunicato esplicitamente dall'utente.

Sostanzialmente i sistemi non utilizzano questi modelli così complessi, basta un modello booleano, strutturale e ce la caviamo, spesso si utilizzano gli stereotipi.

Abbiamo un modello con tantissime caratteristiche come facciamo ad integrarlo... come posso associare un modello ad un comportamento del sistema intelligente se ho tantissimi modelli possibili... non si fa facilmente... quindi uso gli stereotipi.

Lo stereotipo generalizza.

Da quel punto in poi utilizzo il modello stereotipato, semplificato, mi scordo come è fatto l'utente e utilizzo lo stereotipo.

Modellare caratteristiche peculiari dell'utente

A cosa ci interessa la personalità se l'utente interagisce con il pc??? in tanti contesti possono essere importanti..citando di nuovo i sistemi di supporto all'apprendimento intelligente---> docente ad hoc tarato sulle nostre caratteristiche. Il materiale informativo si adatta alle nostre esigenze, alcuni elementi notevoli(personalità stili cognitivi) giocano un ruolo fondamentale.

Modellare il contesto di lavoro

Sebbene il contesto non sempre coincide con dati utili per caratterizzare un utente, viene spesso utilizzato soprattutto nel campo dei *mobile adaptive systems*. Si basa su informazioni osservate relative alle seguenti categorie:

- User Platform: la piattaforma (risoluzione, software, etc) siti adattati per dispositivi mobili, è personalizzato solo perchè utilizziamo un dispositivo mobile...è una personalizzazione banale.

- User Location: Posizione dell'utente. Servizi che l'utente puo' utilizzare, tragitti percorsi, mappe, solitamente gratis dietro ci stanno investimenti. Posso fornire una interfaccia più personalizzata dando informazioni su servizi che l'utente puo' trovare nelle sue vicinanze...esula dal bisogno informativo, è un'informazione fisica.

Ci possono essere altre caratteristiche che potrebbero essere utili...Ambienti che si adattano alle caratteristiche dell'utente...esula dall'ambito web..

Tipicamente si osservano le caratteristiche nel contesto di lavoro corrente e l'adattività avviene individuando lo stereotipo più adatto per caratterizzare l'utente in base alle osservazioni fatte.

Impiego della componente UM nei sistemi di ricerca

La componente di UM puo' intervenire nel processo di ricerca di informazioni in quattro fasi distinte:

1. **All'interno del processo di recupero:** E' il metodo più intuitivo poiche' la componente di UM viene impiegata direttamente all'interno del processo di matching per determinare i documenti più attinenti per l'utente. una volta costruito un modello ci chiediamo a che punto interviene per adattare l'informazione. Parliamo di trattamento dell'informazione.

Il metodo più diretto è un meccanismo di filtraggio in uno stream di informazioni...le info possono essere viste come uno stream...vengono estratte e fornite all'utente. E' tutto un blocco l'utente interagisce con un filtering. Il sistema è molto efficiente. E' tutto cablato nello stesso blocco è molto veloce.

2. **Fase di Re-ranking:** motore di ricerca tradizionale (collezione di molte pagine indicizzate recupero veloce delle informazioni).

Voglio due risultati distinti per utenti distinti...ho ancora il profilo utente...lascio all'utente la possibilità di usare il motore di ricerca, e di sottometergli query pero' i risultati li do in pasto ad un secondo sistema che effettua la personalizzazione..effettua il Re-ranking dei risultati ovvero il motore di ricerca assegna un certo score...che qui viene cambiato in funzione delle informazioni che abbiamo sull'utente.

E' un altro processamento rispetto a quello tradizionale, il tempo per ottenere i risultati aumenta in dipendenza da quanto è lungo il processo di ranking...se voglio riordinare i primi

10 o 100 documenti ho la necessità di rianalizzarli a meno di non avere una copia nel Re-ranking (ma non è così), quindi per effettuare un processo di re ordine dei risultati devo rianalizzare (riscaricare) i documenti dal web. Dei sistemi intermedi invece di riscaricare utilizzano le info fornite dal motore di ricerca (url, titolo, snippet--> estratto del documento che potremmo integrare per rifare re-ranking. potrei analizzare solo quel pezzetto di testo per fare Re-ranking).

3. Fase di Query-modification: Lo UM viene impiegato prima della fase di recupero e selezione dei documenti, attraverso la cosiddetta *query modification*. Potremmo quindi metterlo prima del processo di recupero non personalizzato, Modifichiamo la query facendola tendere di più al profilo che abbiamo dell'utente(modifichiamo la query --> query modification) posso quindi aggiungere parole chiave(augmentation-->sottoprocesso che ricade in query modification) o modificarle. Abbiamo un testo e lo modifichiamo. Se abbiamo un modello di interesse dell'utente possiamo prendere le parole chiave usate dall'utente e aggiungerle a quelle della query e inviarle al motore di ricerca--> semplice diretto, eredita le problematiche viste (vocabolario etc)... riepilogo. E' utile anche per disambiguare. parole chiave che insieme alla parola mi permettono di disambiguarla...metto insieme parola di interesse e query.

E' un metodo molto semplice.

Possiamo pensare anche di utilizzare dei pesi sulle parole.

Modification sostituisco dei termini non totalmente esatti...l'utente non sapeva esprimere il bisogno informativo, posso pensare di modificare la query con dei termini più adatti.

Il problema più grande è che il modello utente è a monte e che la personalizzazione deve passare attraverso il collo di bottiglia della query In più non abbiamo possibilità di modificare il matching (modello a spazio vettoriale la regola del coseno che da i punteggi) possiamo personalizzare solo mediante la query. Gli strumenti sono molto pochi...non posso fare combinazioni infinite di parole chiave...la QUERY E' IL COLLO DI BOTTIGLIA.

4. Recommendation: Sistemi di raccomandazione: Il sistema suggerisce risultati senza l'intervento diretto dell'utente..non c'è la query, l'utente naviga e il sistema fornisce delle informazioni, interviene il profilo utente..Definisco un profilo utente che utilizzo per fornire dati.

Stessi difetti del caso precedente devo passare attraverso il collo di bottiglia motore di ricerca.

Dispensa 7 Tecnologie per la Modellazione Utente

Identificazione e Recupero dei dati sull'Utente

Allo scopo di costruire e mantenere il modello utente aggiornato occorre considerare varie tecnologie:

- **Identificazione dell'utente:** nei sistemi multi-utente e' necessario riconoscere l'utente per associare il relativo modello utente.

- **Recupero dei dati sull'utente:** informazioni, attivita', giudizi e feedback necessari per l'addestramento del modello.

- **Inizializzazione del modello utente:** creazione del modello al momento della registrazione di un nuovo utente.

- **Visualizzazione delle informazioni:** come mostrare i risultati delle attivita' di filtraggio e recupero all'utente.

Identificazione dell'utente

Esistono numerose tecniche per identificare l'utente di un sistema informativo, alcune specifiche per il Web, altre più generiche. Le principali sono:

1. **Cookies:** tag che identificano il computer

Consiste in un testo nella forma chiave-valore memorizzato nel computer dell'utente per mezzo del browser Web. Può essere impiegato dal sito Web per rappresentare varie informazioni, e.g., shopping cart, sessione corrente, preferenze di visualizzazione e memorizzarle lato client. Ad ogni collegamento con il server, il cookie viene restituito dal client al server all'interno del header HTTP. Nel caso dell'identificazione utente, alla 1a interazione col sistema viene rilasciato un codice che identifica univocamente il computer dell'utente nella sessione corrente e nelle successive.

Trasparente all'utente porta però numerosi **svantaggi**: identifica un computer e non un utente. Poco sicuri se non trasmessi con https. Possono essere disabilitati facilmente. Adatti solo per il browsing Web. Valido solo se il client con cui l'utente interagisce col sistema di ricerca è il browser Web. I cookie non sono permanenti: è possibile rimuoverli facilmente, hanno una data di scadenza e il numero di cookie memorizzabile in un browser è limitato (politica FIFO per la cancellazione).

2. **Session ids:** tag che identificano la sessione corrente

Durante una sessione di lavoro, il server Web assegna una chiave che identifica l'interazione corrente. È impiegato spesso all'interno di sessione di e-commerce, per permettere di salvare la shopping cart fino alla fine degli acquisti. Stessi svantaggi dei Cookies

3. **Logins:** username & password

È il tradizionale sistema con form di ingresso. Ogni utente deve ricordare la username e password per identificarsi nel sistema. La password deve essere concepita "*easy to remember, hard to guess*". Il sistema è sicuro ed applicabile in vari contesti (web e desktop). Richiede uno sforzo dell'utente per concepire e ricordare la password.

4. **Software agents:** software ad-hoc da installare lato client

Programmi tipicamente installati sul computer client che per mezzo di protocolli ad-hoc interagiscono direttamente col server. Si ha il vantaggio di poter realizzare interfacce più complesse rispetto alle interazioni rese disponibili dai browser. È possibile inoltre mantenere aperte connessioni per notifiche inviate dal fornitore di servizi all'utente (e.g., nuova news), evitando periodici cicli di polling da parte del client (e.g., protocollo POP posta elettronica). I software agents sono software sviluppati da terzi, perciò esistono questioni sulla sicurezza durante l'installazione sul proprio computer, e sulla privacy nei dati trasferiti. Protocolli ad-hoc potrebbero inoltre non essere compatibili con le network policy.

5. **Proxy servers:** sistemi che monitorano il traffico rete.

Un proxy-server può essere impiegato per valutare le richieste utente e instaurare connessioni tra client e server una volta riconosciuto l'utente. È adatto se la topologia della rete è statica e ogni utente utilizza sempre lo stesso computer. Occorre configurare le applicazioni (e.g., Web browser) per instradare il traffico verso il proxy. Esistono questioni di privacy poiché il proxy instrada le informazioni di vari utenti. Esistono programmi/sistemi

per collezionare il traffico che attraversa un router, invisibili all'utente e senza bisogno di configurare il software lato client.

Recupero dei dati sull'utente

Esistono varie tipologie di informazioni che possono essere raccolte ed analizzate per riconoscere goal, tasks, preferenze, etc., e utilizzate per costruire un UM.

- **user data**: informazioni relative direttamente alle preferenze, bisogni informativi, background, conoscenza etc., dell'utente: query.
- **usage data**: dati estratti monitorando il comportamento dell'utente e l'interazione col sistema
informativo: logs di accesso al server, pagine Web visitate, tracking dei movimenti del mouse o keylogger.

Esistono due approcci per raccogliere queste tipologie di dati:

Explicit user feedback:

la raccolta si basa sull'input diretto dell'utente. Metodo più semplice, dati attendibili (ma occorre tempo e volontà dell'utente). Problema che gli utenti spesso non sanno rappresentare i propri bisogni. L'utente esplicitamente indica al sistema elementi utili per costruire il modello utente. Documenti, musica, video.

Nel dominio dell'Information Retrieval, il *Relevance Feedback* permette all'utente di costruire automaticamente una nuova query a partire dai risultati della query corrente allo scopo di migliorare le prestazioni.

In pratica avviene una fase di *query modification* a partire dal giudizio dell'utente sulle pagine recuperate.

Durante le iterazioni l'utente visita pagine e apprende concetti e keyword per migliorare la rappresentazione dei propri bisogni informativi.

Posiamo pensare di applicare Rocchio al Relevance Feedback:

Rocchio a partire dalla query dell'utente e con i documenti selezionati dall'utente. Abbiamo la query ed il feedback (rappresentato tramite x sui documenti rilevanti per noi) (rilevante concetto con molti significati...rilevante per la query) Possiamo applicare Rocchio: prendiamo il set di documenti rilevati e lo dividiamo in documenti rilevanti per l'utente e tutto il resto saranno documenti non rilevanti(quello che non è stato crocettato dall'utente(senza feedback)) Prendiamo una combinazione opportunamente pesata e riformuliamo la query iniziale. *Partiamo dalla query iniziale e ne creiamo un'altra.* (vedi esempio numerico)

Implicit feedback

la raccolta avviene implicitamente (nascosta).

Feedback implicito. L'utente non interviene direttamente, ma opera come se nulla fosse...

Quello che facciamo è collezionare dati relativi all'interazione tra utente e computer.

Vediamo quali dati sono più utili per costruire un modello utente o no.

Si collezionano dati durante l'interazione dell'utente col sistema (*usage data*) e si estraggono informazioni utili per il modello.

Le categorie di usage-data sono:

- **Browser Cache/History:** pagine ipertestuali visitate durante dall'utente

Le info più facilmente collezionabili sono le pagine visitate. possiamo astrarre la history e utilizzarla. Ci sono dei software più o meno legali che ci permettono di spiare l'utente. tutte le history associano l'url con la data di visita, possiamo vedere quanto tempo l'utente è rimasto su una certa pagina così da attribuire questo tempo ad una misura di interesse...più stiamo su una pagina più è interessante.

Il problema è che è una misura approssimata...e la cronologia è relativa ad una certa postazione. Dovremmo usare la cronologia su tutti i sistemi che utilizziamo.

Ma l'utente non interagisce solo con il Web, anche con doc, mail, etc...questi non possiamo considerarli. Il tempo di visita di una pagina non è sempre attendibile. In una pagina ci sono una serie di elementi che noi a colpo d'occhio filtriamo, ma estraendo solo il testo della pagina l'informazione non strettamente correlata al topic è talmente elevato che il vettore che mi puo' rappresentare questo documento è inutilizzabile..troppo generico o per nulla attinente.

- **Proxy Server Logs:** log di utilizzo di un server proxy

I proxy server possono venire utilizzati per monitorare il traffico Internet generato dai client su una network. In particolare col traffico generato con il protocollo HTTP si controllano le pagine Web visitate. Eventualmente si possono collezionare altre tipologie di traffico, e.g. Messenger, IRC, P2P. Occorre configurare i client per instradare il traffico sul proxy. La cache del browser a volte evita di inoltrare le richieste al server Web e perciò' il proxy puo' perdere una parte di dati..Problemi di privacy.(entra in gioco anche la cache del proxy server. Se la cache del browser contiene una copia ancora valida il client non si connette ed il proxy si perde una parte della comunicazione, più la cache è grande più la comunicazione non avviene.)

- **Desktop Agents:** interazioni con software agents

Software che mi forniscono servizi aggiuntivi. Accedono direttamente all'hd e indicizzano qualsiasi documento applicazione e log...possono accedere a molte informazioni, mettono in un database interno una rappresentazione di questi documenti. Hanno anche copie di cio' che si è fatto e visto in passato, possono accedere alla cache del browser ed alla cronologia.

- **Low level Interactions:** interazioni al livello del sistema operativo

Accedono direttamente ad un livello basso del sistema operativo. Elementi correlati al task di interesse dell'utente. Il problema è che è una info di basso livello (click del mouse, pressione di un tasto..). Ampiezza 100% colleziono proprio tutto, qualsiasi info possibile, il problema è che è a basso livello. Sono eventi alla base di qualsiasi attività, task, ma sono di astrazione troppo bassa che faticiamo a tirarne fuori qualcosa di utile). Non si riesce a salire di livello ed interpretare queste azioni per arrivare a capire il goal.

- **Web Server Logs:** log estratti dai server Web

Log dal lato server. Info molto utile, (è l'informazione di history del browser pero' lato server) ho anche il tempo (quindi considerazioni viste prima)

Però' contiene tutte le richieste di tutti i client, il server non fa un log per ogni client...bisogna riconoscere il traffico per un certo utente allora tecniche di User tracking. Problemi: Presenza

di NAT o il fatto che l'utente puo' accedere ad un server con un indirizzo ip non suo(mappato su un altro) indirizzo locale diventa pubblico usato da più utenti...quindi confusione su quale utente è associato a quale indirizzo ip. Se vogliamo fare un tracking dell'utente, o facciamo log password cooking, (lato cliente). Oppure facciamo delle uristiche. Due utenti che utilizzano la stessa versione di browser con gli stessi plug-in..mappo quindi ip e agent filed con un certo utente.

Session Timeouts diamo un limite a due eventi. In generale troncane le sessioni troppo lunghe (perchè potrebbe essere un agent software o l'utente dorme) sono euristiche non sicure che vengono utilizzate per utenze server dove non c'è il log in..etc...

- **Query Search Logs:** query sottomesse dall'utente

log delle query sottomesse. Utilizzandole per creare profili utente. Molto utili e contengono pochissimo rumore. Se l'utente è avanzato va a vedere la pagina e si ferma lì, se l'utente è inesperto torna indietro e riformula la query. Abbiamo i risultati visitati e quelli ignorati. C'è il vantaggio di valutare l'informazione usata per la ricerca e fornire personalizzazione istantanea. Occorre però che l'utente si registri sul sito. Inoltre l'utente puo' sottomettere query a più motori di ricerca e non è possibile collezionarle tutte contemporaneamente.

Tassonomia:

Ci permette di categorizzare le azioni. Sono tutte implicite (tranne forse rate) non c'è mai una azione esplicita per un motore di ricerca per dire:"ok questo mi interessa"

Dobbiamo sapere che esiste una certa categorizzazione e possiamo associare le azioni ad un meccanismo di profiling dell'utente...

E' possibile catalogare il tipo di feedback implicito che e' possibile riscontrare durante le attivita' dell'utente in base a due caratteristiche:

Behavior Category: *lo scopo relativo alle attivita' dell'utente.*

Minimum Scope: *l'ambito di azione delle attivita' utente*

Considerazioni:

L'implicit feedback e' una tecnica fondamentale per collezionare dati utili in modo trasparente, senza che l'utente venga coinvolto e debba impiegare risorse di tempo e cognitive per interagire col sistema informativo.

- Meno tempo per la ricerca, piu' tempo per le informazioni.

Gli usage data contengono spesso del rumore, cioe' informazioni poco attinenti i bisogni informativi, perciò occorre processare e filtrare i dati.

Catturare le attivita' dell'utente spesso richiede l'installazione di software ad hoc sul computer dell'utente.

Le questioni sulla privacy rimangono un tema aperto.

Per quanto riguarda il confronto con le tecniche esplicite, in fatto di performance ci sono studi contrastanti.

Tecnologie per la modellazione dell'Utente

Inizializzazione del modello Utente

Quando un nuovo utente si registra nel sistema viene costruito un nuovo profilo utente. Inizialmente tale profilo è vuoto e perciò la personalizzazione dei contenuti e dell'interazione non può avvenire. È importante inizializzare il profilo il prima possibile.

A tempo zero il modello utente è vuoto, Ci sono dei metodi che a startup possono riempire il modello utente

Tecniche di explicit e implicit feedback impiegano tempo per collezionare ed analizzare dati riferiti all'utente. Per tale motivo esistono metodi per inizializzare profili:

- **Manual:** Nell'approccio manuale il sistema informativo richiede all'utente di fornire direttamente dati relativi ai propri interessi oppure informazioni utili allo scopo di riconoscerli.

E' una forma di feedback esplicito che viene operata al momento della registrazione di un nuovo utente. Per tale motivo soffre degli stessi svantaggi, e.g., carico cognitivo e tempo impiegato dagli utenti. *Svantaggio:* l'utente deve fare questa intervista che non serve per il suo bisogno informativo, sono attività estranee e superflue per l'ottenimento della conoscenza. Ad ogni domanda c'è un carico cognitivo da fare.

Es: Google Personalized Search: Gli interessi possono essere scelti per mezzo di categorie e sottocategorie. Probabilmente delle parole chiave associate a queste categorie venivano associate alla query. La scelta era solo booleana...non si capisce la semantica dell'azione per l'utente. Nel prototipo di Google e' possibile anche variare il livello di personalizzazione operato durante il recupero dei documenti. E' importante allo scopo di evitare il fenomeno del *Tunnel Vision*:

A causa della presenza del profilo utente basato sugli interessi, i sistemi di filtraggio e recupero personalizzato tendono a fornire informazioni simili a quelle viste in passato e giudicate di interesse. Non siamo sicuri che il modello utente rappresenti sempre in maniera corretta l'utente.

Esempio banale: nuovo paradigma di programmazione, nel nostro modello di interessi non è presente (è uscito ora) non l'abbiamo ancora cercato, il profilo utente non ha ancora avuto tempo di aggiornarsi, ma a noi ci interessa...il problema di filtraggio non ce lo propone perchè non risulta nel nostro modello, ci perdiamo tutto quello che succede intorno a noi. In alcuni domini è molto importante: adaptive systems.

Molte informazioni vengono quindi ignorate se pur ci interessano...si è quindi pensato di regolare il modello di personalizzazione per regolare se ci interessano cose nuove oppure gli interessi non sono variati. Riducendo la capacità di personalizzazione e' possibile aumentare le chance di diversificare le informazioni presenti nei risultati

- **Stereotypes:** Tecnica molto importante perchè permette di rappresentare eventi e oggetti che occorrono in una certa rappresentazione con descrizioni parziali.

Permette di fare predizione sui relativi interessi, o ha delle caratteristiche utili per filtraggio, raccomandazione. Si può definire uno stereotipo come "un mezzo per rappresentare descrizioni parziali di eventi o oggetti che occorrono frequentemente". La stereotipizzazione degli utenti può essere vista anche come un problema di categorizzazione (quale classe e' più rappresentativa per un certo utente?).

Gli Stereotipi sono impiegati solitamente quando si verificano le seguenti ipotesi:

- Esiste un gran numero di utenti con caratteristiche eterogenee e gli utenti sono distribuiti omogeneamente in base a tali caratteristiche.

- Il funzionamento del sistema e' basato su un numero elevato di possibili personalizzazioni. Se abbiamo da adattare semplicemente un'interfaccia lo stereotipo non serve. Serve quando ci sono molte info molti controlli...molto.

Esempi: bibliotecario/sistema automatico di ricerca che deve suggerire un libro (deve acquisire alcune conoscenze su chi glielo chiede per suggerirgli quello più adatto).

Sistemi basati su stereotipi hanno perciò una *base di conoscenza* impiegata per istanziare profili dell'utente, mentre sistemi basati su altri approcci devono creare il modello iniziale con tecniche solitamente più onerose perché richiedono di collezionare molte più informazioni esplicitamente.

Uno stereotipo rappresenta le caratteristiche (o *facets*) condivise da una classe di utenti. Le *facet* dipendono ovviamente dal dominio o task che si sta studiando. Tipicamente ad ogni *facet* è associato un valore appartenente ad una certa scala.

Il compito di un sistema basato su stereotipi è riconoscere gli eventi che sono associati ad un particolare stereotipo.

Si definiscono un insieme di eventi (o *trigger*) il cui verificarsi viene associato a particolari stereotipi che possono venire *attivati*.

Quando si attiva uno stereotipo, le predizioni ad esso associate vengono incluse nel modello utente, e.g.:

- Se il Trigger è la Scelta di un comando avanzato
- Allora al Modello utente si associa lo stereotipo Utente professionale

Può accadere di includere informazione errata all'interno del modello utente, ad esempio perché si è scelto lo stereotipo sbagliato. Occorre includere nel modello anche una misura di confidenza (o *rating*) circa le informazioni incluse che indica quanto il sistema le ritiene attendibili.

Uno stereotipo può essere perciò definito come un insieme di triple:

- (attribute, value, rating)

Al verificarsi di più trigger può accadere di trovarsi in situazioni di conflitto tra stereotipi:

- Molti UNDO + scrittura veloce -> utente principiante o esperto?

Per tale motivo solitamente si tiene traccia anche delle motivazioni (o *justification*) che hanno portato ad una certa inferenza e si assegna un rating anche allo stereotipo associato ad un certo trigger. (Vedi esempio Grundy).

Grundy

Lo stereotipo impiegato in Grundy è composto da insiemi di triple:

- *facet*: nome della caratteristica
- *value*: con un valore nel range [-5...+5]
- *rating*: con un valore nel range [0-1000] con 1000 pari a massima certezza (ogni valore dipende dalla classe dell'utente).

Tipicamente uno stereotipo contiene un numero limitato di *facet* rispetto a quelle possibili, e a quelli contenuti in altri stereotipi.

Grundy si organizza gli stereotipi in modo gerarchico (un grafo diretto aciclico)...via via stereotipi sempre più specifici che affinano le caratteristiche generali iniziali.

Cerchiamo sempre di dare uno stereotipo generico (ANY-PERSON). Ogni volta che ci accorgiamo che l'utente si discosta dalla classe cerchiamo un altro stereotipo.

Un trigger viene rappresentato da una tripla:

- *stereotype*: nome dello stereotipo da attivare
- *rating*: numero compreso in [0, 1000] da assegnare allo stereotipo
- *reasons*: motivazione che ha scaturito tale trigger.

Il rating all'interno del trigger rappresenta quanto il relativo stereotipo da attivare è adatto alla particolare situazione che ha scatenato il trigger. Generalmente ad un trigger corrisponde uno stereotipo.

Ogni stereotipo ha associata la lista di trigger che lo hanno attivato.

L'attivazione di un trigger e' influenzata dal fatto che esso sia stato gia' attivato in passato:

- Se e' gia' stato attivato in passato, non accade nulla.
- Se non e' stato attivato, il corrispondente stereotipo viene attivato

Anche in questo caso, l'attivazione di uno stereotipo dipende se lo stereotipo e' gia' stato attivato in passato:

- Se e' gia' stato attivato, ed e' correntemente attivo, il valore dei ratings dello stereotipo e dei

trigger vengono incrementati per riflettere la situazione in cui c'e' stata una conferma sullo

stato di attivazione.

- Se lo stereotipo non e' piu' attivo, la situazione deve essere riesaminata in base alle nuove informazioni per capire se attivarlo o meno.

Uno stereotipo puo' avere un valore associato ad una facet elevato ma il trigger che lo ha attivato puo' assegnarli un rating basso poiche' la situazione non garantisce che lo stereotipo sia quello piu' corretto.

Si puo' ad esempio rapportare il valore del facet in base al valore del rating dello stereotipo. (valore facet * valore stereotipo dato dal trigger /1000)=valore facet

Attivazione stereotipi:

Un trigger viene attivato in tre situazioni:

- All'interno del task impiegato per ottenere informazioni dall'utente.
- Nel caso in cui venga assegnato un valore ad una facet. Ad esempio, nel caso in cui al facet

gender venga assegnato il valore male, sara' automaticamente attivato lo stereotipo MAN.

- Nel caso in cui uno stereotipo venga attivato, e' possibile che un trigger associato ad esso venga attivato a sua volta. E' utile nel caso si voglia assegnare all'utente piu' stereotipi a partire

dall'attivazione di uno in particolare. Ad esempio, l'attivarsi dello stereotipo *scientist* potrebbe

attivare il trigger che attiva a sua volta lo stereotipo *atheist*.

Conflitti sui valori, potrei avere delle considerazioni discordanti (perché ho più stereotipi attivati). Facciamo quindi una media delle confidenze: peso modulato in base alla confidenza. Costruire una base di conoscenza per gli stereotipi è oneroso, eredito le caratteristiche che già conosco.

Se mi accorgo che gli stereotipi son corretti mi fermo altrimenti cerco di adattare il più possibile l'informazione che ho. Se faccio molti errori il valore delta (che altera la confidenza dei valori correnti)avrà un valore elevato (inizialmente è un valore elevato). Quando ho molte info tendo a fare alterazioni più piccole(delta minore, influenza meno il sistema).

- Training Set:

Nel meccanismo di inizializzazione basato su un insieme di elementi di training, l'utente segnala degli esempi che possono venire utilizzati dal sistema per costruire il profilo.

Ad esempio, in un sistema di ricerca sul Web l'utente puo' segnalare un feedback su dei documenti giudicati di interesse.

Per mezzo di tecniche di IR (vedi Rocchio) o machine learning (classificatori bayesiani naive) e' possibile determinare un profilo iniziale degli interessi.

Ha il vantaggio di essere simile ai meccanismi di relevance feedback impiegati normalmente nei sistemi di recupero, ma lascia all'utente la scelta dei documenti di interesse. Questo può portare ad indicazioni poco precise.

Content-based Filtering, Collaborative Filtering, Focused Crawling (visualizzazione delle informazioni)

Content-based Filtering:

Nell'approccio Content-based Filtering, il sistema informativo filtra le informazioni disponibili con rappresentazioni basate sul contenuto dei documenti (testo, audio, etc). L'operato di un utente non viene influenzato dal comportamento degli altri utenti del sistema ma solo dal suo operato (non è sempre vero, ma due utenti che interagiscono nel sistema non si influenzano, non è un approccio collaborativo).

Corrisponde all'approccio impiegato nelle tecnologie e sistemi visti fin'ora.

Gli svantaggi di questo approccio sono:

La costruzione delle rappresentazioni delle risorse richiede spesso di analizzare contenuti testuali in linguaggio naturale (e.g., pagine Web). Solo rappresentazioni sofisticate possono affrontare il problema del vocabolario (e.g., associare il giusto significato ad ogni termine). Rappresentazioni più dirette basate direttamente sulle keyword possono creare falsi match. Per alcune risorse non è possibile associare una rappresentazione chiara esatta del contenuto. Dobbiamo aggiornare il modello nel tempo.

Collaborative Filtering:

L'approccio collaborativo punta a risolvere parti di questi problemi.

Nell'approccio di Collaborative filtering (CF) la selezione delle risorse è basata sulle opinioni espresse da altri utenti del sistema.

In tale approccio si assume che le persone con gusti simili condividano gli stessi pareri sulle risorse disponibili.

- Nell'approccio Content-based invece si assume che per oggetti con caratteristiche simili vengano espressi gli stessi giudizi da parte di un utente.

Si ispira al comportamento usuale delle persone di scambiarsi pareri prima di esprimere un giudizio, acquistare, scegliere una strategia.

Es: Sistema di raccomandazione per film.

Il rating mi serve per capire in questo sistema quali utenti sono simili a me.

Lista di film che non ho visto ordinata in base alla predizione che fa il sistema sul nostro modello utente. Ad utenti simili a te con le preferenze è piaciuto molto. Come rappresentare questi giudizi: in forma matriciale.

Esistono due tipologie di rating: Espliciti (espressi direttamente dall'utente) o

Impliciti (collezionati automaticamente dal sistema in base alle azioni dell'utente).

Esistono due principali algoritmi non-probabilistici:

- *User-based Nearest Neighbor*: la predizione suggerita ad un utente dipende dai giudizi di utenti simili. Come possiamo definire la similarità tra due utenti...basandosi sugli stessi rating..confronto coppie di rating se vedo che tutti coincidono ho un perfect rating...se tutti sbagliano ho un disagreement, altrimenti non ho correlazione..

Calcoliamo la distanza in funzione del valor medio poiché Un utente u può assegnare in media valori molto alti mentre un utente n valori in media bassi. Il valore medio dei ratings $\{r\}$

permette di confrontare i rating dei singoli item ignorando tali discrepanze. Per questo motivo normalizziamo i valori. Dato un insieme di valori vede se la distribuzione dei valori è simile o no ad un'altra

- *Item-based Nearest Neighbor*: la predizione su un item e' basata sui rating espressi su item simili. Approccio duale dove il valore di predizione non dipende dagli utenti simili a me...E' più complesso ma funziona meglio. La predizione è dovuta agli altri items che hanno una distribuzione simile... Prendo gli items che hanno avuto gli stessi valori dagli utenti.

Utenti diversi hanno dato valori simili.

Unica accortezza sono utenti che hanno espresso pareri su item distinti (non in comune come prima). Questo algoritmo da prestazioni migliori.

Problemi: Esistono anche altri approcci/algoritmi.

Problema molto importante è che a tempo t_0 non ho info non ho rating...non posso fare confronti tra utenti o items (**cold start**) problema se arriva un nuovo utente o nuovo item.

Come si può scalare un algoritmo a milioni di items e utenti? Ridurre le "dimensioni" possibili; Processamento offline per individuare preventivamente gli utenti (o item) più simili tra loro in modo che la computazione online sia più veloce.

Information Filtering e il Web

Il processo di Information Filtering opera tipicamente su flussi di informazioni (e.g., RSS news). In alcuni casi, sebbene tali informazioni vengono create ed alterate nel tempo, non sono ordinate in modo temporale. Nel caso del Web, le pagine Web sono memorizzate su server Web dislocati ovunque. Un sistema di filtraggio non ha accesso diretto a tali informazioni se non visitandole una alla volta.

Per garantire l'accesso e il filtraggio di tali informazioni si può pensare di utilizzare delle collezioni già esistenti, e.g., gli indici dei motori di ricerca (vedi approccio di Re-ranking), oppure impiegare dei crawler che visitano il Web.

Data l'ampiezza del Web non è possibile in breve tempo avere accesso a tutte le pagine esistenti. Occorre limitare la ricerca alle pagine che più verosimilmente interesseranno l'utente o un gruppo di utenti (e.g., gruppi di ricerca, aziende, traders in borsa).

Focused Crawling

I focused crawler sono dei crawler che gestiscono l'ordine delle pagine da visitare in base a quanto esse possono essere d'interesse per l'utente. In pratica, il loro obiettivo è decidere quali sono i percorsi migliori.

Vantaggi:

Riduzione risorse di rete (si scaricano solo pagine utili) riduzione memoria necessaria

Si ignorano le pagine di non interesse (aumento del coverage) pagine scaricate giudicate di interesse.

Aumento della freshness a tempo che intercorre tra il download e l'analisi da parte del sistema di recupero/filtraggio.

Più facile aumentare la precision nei risultati del filtraggio.

Nella pratica a partire da un insieme iniziale di pagine (*seed set*) il focused crawler deve stabilire l'ordine delle prossime pagine da visitare. Tali pagine sono ottenute a partire dai link del seed set.

Durante la ricerca il seed set cresce in dimensione. Alla fine dell'esplorazione esso corrisponde al risultato dell'esplorazione che puo' venire analizzato da un sistema di filtraggio.

Le informazioni disponibili al crawler sono:

- Contenuto testuale delle pagine visitate
- Le ancore testuali dei link nelle pagine
- Il grafo composto dalle pagine Web e dai link che le uniscono.

Le tecniche maggiormente utilizzate si basano sul fenomeno della *topical locality*:

- Una pagina relativa ad un certo topic verosimilmente contiene link verso pagine relative allo stesso topic.

Inoltre e' possibile utilizzare particolari algoritmi sul grafo $G=(V,E)$ dove i vertici corrispondono alle pagine e gli archi ai link che le uniscono, per determinare pagine di maggiore rilevanza (chiamato processo di *distillazione*) per mezzo di misure di "popolarita":

- Se una vertice (pagina) ha molti archi entranti e' piu' importante rispetto ad altre.