Indice

Capitolo 1

Introduzione

Negli ultimi anni internet è cresciuto notevolmente, basti pensare che si è passati dai 16 milioni di utenti del 1995 ai 1971 del 2010. Questo sviluppo ha comportato un incremento costante della quantità di siti presenti nella rete come dimostrano le seguenti statistiche[1]:

- 255 milioni – il numero di siti web al Dicembre del 2010
- 21,4 milioni – siti aggiunti nel 2010
- 1,97 miliardi – utenti internet nel mondo nel Giugno del 2010
- 14% – aumento degli utenti internet nel 2010 rispetto all'anno precedente

Sulla base di questi dati risulta quindi evidente l'importanza dei motori di ricerca per un utente nel ricavare informazioni senza perdersi nella vastità di internet. Nonostante i continui sforzi per migliorare le tecniche di ricerca, ancora oggi può accadere che i risultati non risultino soddisfacenti o che quelli nelle prime posizioni non siano pertinenti.
La domanda che ci si è posti è se fosse possibile migliorare l'esperienza dell'utente nella ricerca di informazioni presentando risultati migliori data una query. Ciò che si vuol fare è combinare un motore di ricerca con una piccola rete di utenti; un utente ha molti amici e se si fida di questi la loro opinione può essere utilizzata per modificare i risultati.

L'idea di utilizzare una rete di utenti all'interno di un motore di ricerca è maturata proprio osservando lo sviluppo del web negli ultimi anni e quindi la nascita di tutti quei siti chiamati Social Network.
Alcuni di questi sono presenti nella vita di tutti i giorni delle persone; basta guardare Facebook con la maggior parte degli utenti che si collegano quotidianamente per condividere informazioni(foto, stati, link, ecc.) o semplicemente per tenersi in contatto con i propri amici.

[1] http://computeriamo.com/2011/01/31/statistiche-internet-per-il-2010/

Facebook è solo la cima di una montagna formata da un numero elevato di social network tra cui i più importanti e utilizzati sono:

- YouTube
- Twitter
- Flickr
- Digg
- LinkedIn
- Google+

Riportiamo alcune statistiche[2] riguardo ai social network in modo tale da capire la loro importanza e diffusione:

- SOCIAL MEDIA
  - 152 milioni – il numero di blog
  - 100 milioni – nuovi account su Twitter nel 2010
  - 25 miliardi – tweet nel 2010
  - 600 milioni – persone su Facebook alla fine del 2010
  - 250 milioni – nuovi utenti registrati su Facebook nel 2010
  - 30 miliardi – contenuti (link, foto, ecc.) condivisi su Facebook al mese
- VIDEO
  - 2 miliardi – numero di video guardati su YouTube ogni giorno
  - 35 – ore di video caricate su YouTube ogni giorno
  - 20 milioni – video caricati su Facebook ogni mese
- IMMAGINI
  - 5 miliardi – foto su Flickr al 2010
  - 3000 – foto caricate al minuto su Flickr
  - 3 miliardi – foto caricate su Facebook al mese

La stessa soluzione è stata pensata e avviata da Benvenuti Stefano, il quale ha dato vita al progetto MilkRain[1]. Come detto sopra il sistema da lui sviluppato si basava sulla combinazione di un motore di ricerca e una rete di utenti, i quali condividevano risorse e valutazioni in modo tale da raffinare i risultati di una ricerca.
Il sistema risulta essere basato sul concetto di trust[2] tra gli utenti, e cioè una risorsa può essere raccomandata ad un utente non solo per il suo contenuto ma anche considerando il profilo dell'utente o la fiducia tra quest'ultimo e un altro utente che raccomanda la risorsa.

Prima di vedere come è strutturato MilkRain vediamo quali sono i concetti principali alla base del progetto.

[2] http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/

1.1 Recommender System

Le informazioni riguardo ad un utente possono essere utilizzate per prevedere la valutazione (in termini di apprezzamento) che quest'ultimo potrebbe dare ad una risorsa di qualsiasi tipo come ad esempio un film, un programma TV, musica, libri, immagini, ecc. Un sistema che realizza questo si chiama Recommender System.
Solitamente, un recommender system confronta il profilo utente con alcune caratteristiche di riferimento e cerca di capire quale sarà il "voto" che l'utente potrebbe dare ad un item che non ha ancora considerato. Queste caratteristiche possono essere la similitudine nel contenuto dell'item (content-based filtering) o l'apprezzamento da parte di altri utenti dell'item (collaborative filtering)[3].

Un sistema basato sul content-based filtering seleziona gli item in base alla correlazione tra il contenuto di questi e le preferenze dell'utente, al contrario del collaborative filtering che sceglie item basati sulla correlazione tra persone con simili preferenze. Vengono presentati all'utente solo gli item la cui correlazione risulta essere superiore a una soglia preimpostata.

Proviamo ad esempio a considerare un sistema che tiene traccia di documenti che vengono letti online. Il contenuto di un documento può essere rappresentato da un insieme di parole. Queste vengono estratte dai documenti attraverso dei passi; come prima cosa tutti i tag HTML e le parole che vengono usate più frequentemente vengono eliminate (in quanto non possono essere usate come discriminatori), successivamente le rimanenti parole sono ridotte alla radice, eliminando prefissi e suffissi.
Il profilo dell'utente in questo esempio è rappresentato sempre da parole, ma quelle dei documenti che l'utente ha trovato interessanti. I documenti che l'utente ha trovato interessanti sono determinati usando dei feedback che possono essere impliciti o espliciti.
I primi consistono nell'osservare le azioni dell'utente, nel nostro esempio:

- Click del mouse
- Dove si ferma il mouse
- Tempo trascorso
- Selezione di una pagina (libri, news, ecc)
- Frequenza delle azioni

è più conveniente ma è più difficile da implementare; mentre nei feedback espliciti si richiede che l'utente dia delle valutazioni sull'item appena visto:

- Domande sì/no
- Domande a risposta multipla
- Domande aperte

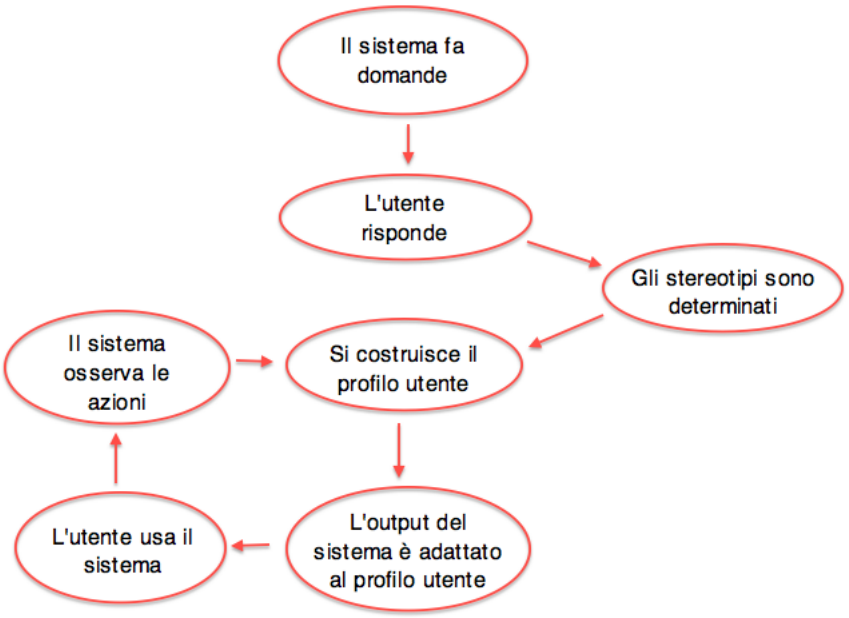

Figura 1.1.1: Sistema di raccomandazione

Ci sono diversi modi con i quali rappresentare i termini che verranno usati per nella fase di aggiornamento del profilo utente[4]. Un metodo di rappresentazione spesso usato è il vector space model. In questo modello un documento D è rappresentato da un vettore m-dimensionale, dove ogni dimensione corrisponde a una distinta parola ed m è il numero totale di parole presenti nella collezione dei documenti. In ogni cella del vettore si scrive il peso wi del termine ti che indica la sua importanza. Se il documento D non contiene il termine ti allora il suo peso wi sarà zero. I pesi dei termini possono essere determinati utilizzando la tecnica tf-idf (term frequency–inverse document frequency) che assegna i pesi in base a quanto spesso appaiono in un documento e quanto frequentemente compaiono nell'intera collezione dei documenti:

$$w_i = tf_i \times \log\left(\frac{n}{df_i}\right) \tag{1.1}$$

dove tfi è il numero di occorrenze per il termine ti nel documendo D, n è il numero totale di documenti nella collezione e dfi il numero di documenti nei quali il termine compare almeno una volta.
Nel vector space model i profili degli utenti possono essere rappresentati come i documenti da uno o più vettori profilo. Il grado di similarità tra un documento D e un vettore profilo P, dove P = (u1,..., uk) può essere determinato tramite la formula:

$$sim\,(D,\,P) = \frac{D \cdot P}{\|D\| \cdot \|P\|} = \frac{\sum_k u_k \cdot w_k}{\sqrt{\sum_k u_k^2 \cdot \sum_k w_k^2}} \tag{1.2}$$

Il content-based filtering ha però una serie di problemi da non trascurare[5]:

- bisogna conoscere il contenuto degli item (richiede una indicizzazione manuale o automatica)
- le caratteristiche degli item non "catturano" tutto
- "user cold-start" (bisogna apprendere quali sono i contenuti importanti per un utente, questo richiede tempo)
- serendipity (l'utente potrebbe scoprire un item di suo interesse che non ha nulla a che fare con quelli visualizzati di solito)

Per limitare in parte questi problemi si può utilizzare un altra tecnica, il collaborative filtering. In questo caso per prevedere se un item può interessare all'utente il sistema utilizza le opinioni degli altri utenti. Proprio per questo è un filtraggio collaborativo.
Rispetto al content-based filtering con questa tecnica non è necessario analizzare il contenuto degli item, si possono catturare delle caratteristiche più nascoste e il fenomeno della serendipity è meno presente.
Esistono due tipi di collaborative filtering: user-based collaborative filtering e item-based collaborative filtering.
Nel primo per prevedere l'opinione di utente per un item si usa l'opinione degli utenti simili a quello in questione. La similarità fra gli utenti viene decisa cercando delle "sovrapposizioni" nelle valutazioni date ad altri item.
Anche in questo caso, come per il content-based filtering, è presente il problema dello user cold-start problem (se l'utente è nuovo non abbiamo abbastanza conoscenze per sapere chi è simile a lui). Altri problemi sono la sparsità (quando si fanno raccomandazioni per un grande insieme di item c'è il rischio che un utente ne abbia valutati pochi e quindi risulta complicato individuare degli utenti simili), la scalabilità (con milioni di utenti e item le computazioni diventano lente).

Nell' item-based collaborative filtering l'idea base è che un utente potrebbe avere la stessa opinione per item simili, come nel content-based filtering, ma in questo caso la similarità è decisa guardando come altri utenti li hanno valutati. Se lo confrontiamo con lo user-based collaborative filtering ha il vantaggio di prevenire lo user cold-start problem, inoltre migliora la scalabilità.
Un problema invece che si aggiunge è questa volta l'item cold-start, non posso prevedere quali item sono simili ad un altro fino a quando non ho dei giudizi per questo da parte degli utenti.

1.2 Data Mining e User Profiling

Ciò che si vuole è avere a disposizione un software che racchiuda tutti i diversi profili virtuali di una persona sul web. L'utilità principale risiederebbe sicu-

ramente nel fatto che per un utente esterno è molto semplice individuare una persona e osservare i suoi "movimenti" in internet a partire dalle sue interazioni con altri utenti fino a capire quali sono gli interessi della persona in questione. Quello che offre questa applicazione è quindi una grande mole di dati relativi a un numero altrettanto elevato di utenti; ed è proprio qui che entra in gioco il concetto di Data Mining.

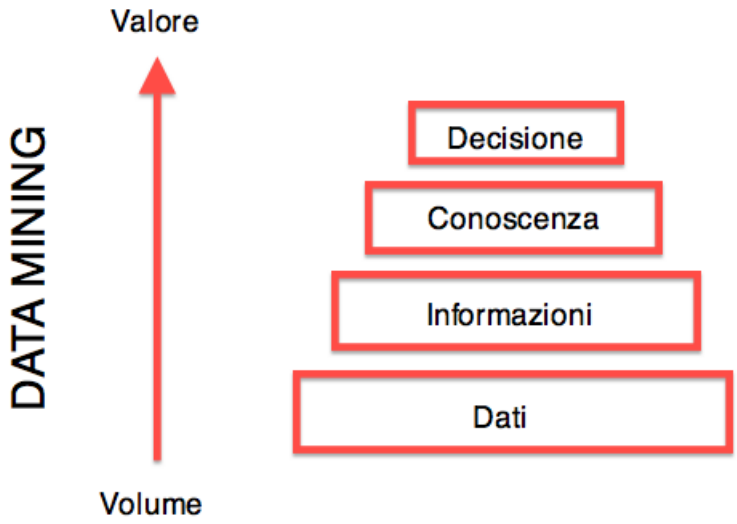

Figura 1.2.1: Schema Data Mining

Il Data Mining è il processo di estrazione di conoscenza, in termini di informazioni significative ed immediatamente utilizzabili, da grandi moli di dati, tramite l'applicazione di particolari tecniche ed algoritmi.
Avere i dati non è un problema, il difficile arriva quando questi dati devono essere utilizzati per estrarne delle informazioni; infatti nella maggior parte dei casi i dati si presentano in forma non strutturata o non nel modo adatto per ricavarne informazioni utili.
Se vogliamo realizzare un progetto come il nostro che richiede l'applicazione di tecniche di Data Mining, dobbiamo realizzare diverse fasi per raggiungere l'obiettivo desiderato.
Come consigliato da Cineca[6], il percorso da seguire dovrebbe appunto essere:

- Individuazione delle fonti di dati
- Estrazione / acquisizione dei dati (ed integrazione, se provenienti da fonti o data bases diversi)
- Pre-processing (Pulizia dei dati - Analisi esplorative - Selezione - Trasformazione - Formattazione)

- Data Mining (Scelta dell'algoritmo - Individuazione dei parametri - Elaborazione - Valutazione del modello)
- Interpretazione / valutazione dei risultati
- Rappresentazione dei risultati

Pensiamo al nostro applicativo che raccoglie tutti i diversi profili di una persona nel web, o anche solo delle utenze di Facebook, è complicato da queste capire quali sono le relazioni tra diversi utenti, quali sono gli interessi di questi, quali prodotti potrebbere piacere ed essere acquistati.
Detto questo si può benissimo intuire che l'uso di questo applicativo potrebbe essere non solo effettuato da un privato, come abbiamo fino ad ora illustrato, ma potrebbere essere sfruttato soprattutto dalle aziende che si interessano di vendere prodotti di qualsiasi tipo.
Facciamo un semplice esempio: un utente X registrato su Facebook navigando sul social network finisce nella pagine relativa a una marca di scarpe. L'utente clicca sul tasto "Mi Piace" e automaticamente questa informazione viene registrata sul profilo dell'utente ed è visibile a tutti, anche dalle aziende che possono mandare sulla pagina dell'utente X delle pubblicità mirate proprio relative a quella marca di scarpe.

Ogni utente di Facebook è valutato fino a 100 dollari e quindi vale ufficiosamente 50 miliardi di dollari per gli investitori che sono la banca Goldman Sachs ma anche la Microsoft, il miliardario russo Yuri Milner e il magnate di Hong Kong Li Ka Shing. E' a loro che gli utenti di Facebook stanno consegnando i loro profili personali completi di età, indirizzo, data di nascita, sesso, gusti; in cambio di migliaia di servizi gratuiti, dai giochi agli incontri amorosi. Mentre un utente gioca o clicca "mi piace" sulla pagina di una marca di scarpe piuttosto che su quella di un profumo, altri si comprano e vendono il nome e cognome degli utenti, i loro gusti e la loro lista di amici.
Sono i dati degli utenti il vero tesoro; infatti si è scoperto che i giochi della Zynga Farmville, Frontierville, Mafia Wars, Treasure Isle, ma anche i quiz e varie altre applicazioni presenti su Facebook trasmettono l'identificativo degli utenti all'esterno, ad altre aziende che compilano profili e li rivendono ai pubblicitari[7].

Ed è proprio qua che entra in gioco un altra parte molto importante e collegata al Data Mining, sto parlando dello User Profiling.
Lo User Profiling è l'attività per mezzo della quale una serie complessa di dati relativi ad utenti o clienti viene elaborata da specifici programmi al fine di generare la segmentazione della propria utenza in gruppi omogenei di comportamento. I dati che possono essere presi in considerazione per la profilazione sono molteplici[3]:

- la serie delle scelte di navigazione effettuate sul sito in esame dagli utenti unici identificati;
- la dichiarazione esplicita di preferenze e interessi ottenuta tramite procedure di registrazione o sondaggi;

[3]http://www.diodati.org/scritti/2002/g_stat/stat05.asp#mining

- la raccolta di dati demografici;
- la risposta degli utenti identificati a promozioni o a contenuti particolari.

La cosa fondamentale è che questo procedimente di creazione di profili degli utenti istituisce delle correlazioni tra i dati raccolti che consentono di ricavare informazioni utili per un utente privato oppure, cosa più importante, commercialmente utili per le aziende di qualsiasi tipo.
Esempi di correlazioni tra i dati sono ad esempio:

- Content affinities (affinità di contenuto) – gli insiemi di contenuti che tendono ad essere visti insieme dagli utenti del sito esaminato;
- Content effectiveness (efficacia dei contenuti) – per i siti di commercio in Rete, i contenuti che tendono ad essere visti in sessioni-utente che si concludono con un acquisto;
- Product affinities (affinità di prodotto) – sempre per i siti di commercio in Rete, l'elenco dei prodotti che sono più spesso acquistati insieme.

La profilazione di un utente è fatta dalla maggior parte dei siti internet. Un semplice ma allo stesso tempo molto esplicativo esempio lo si può avere se un qualsisi utente registrato su google digita nel browser https://www.google.com/dashboard.

A questa pagina, chiamata appunto cruscotto, si possono vedere tutte le operazioni svolte dall'utente in un arco di tempo con le applicazioni dell'azienda di Mountain View, e quindi tutto quello che Google conosce dell'utente. Proviamo a pensare a quanto semplice sarebbe incrociare i nostri dati con quelli di altri utenti e immaginiamo quante informazioni avrebbero su ogni singolo utente.
Mentre alcuni siti, come appunto Google, devo costruirsi un profilo a seconda delle informazioni che gli utenti mandano, nei social network sono gli utenti stessi a costruire un profilo di loro stessi. Resta comunque un profilo con informazioni abbastanza "grezzi", ed è proprio per questo che il nostro software, una volta raccolti questi dati dovrebbe realizzare una profilazione molto più semplice e con informazioni più mirate di ogni utente, ottimizzando quindi anche i tempi di ricerca delle informazioni.

1.3 MilkRain

MilkRain consiste in un reccomender system per contenuti web basato sulla combinazione di un motore di ricerca con una semplice rete sociale nella quale gli utenti condividono tra loro dei topic in comune.
Riportiamo di seguito una sintesi del funzionamento del sistema creato da Benvenuti e successivamente verranno esposte le modifiche che si vuole apportare al progetto con questa tesi.

Moduli principali

In questa sezione verrà riportata una sintesi del funzionamento di MilkRain; per un approfondimento rimandiamo a [1].

Il progetto è suddiviso in moduli in modo tale da rendere più semplice aggiungere nuove caratteristiche nei lavori futuri. I moduli che compongono il sistema sono:

- una rete di utenti
- data sources
- un classificatore di query
- combine
- feedback

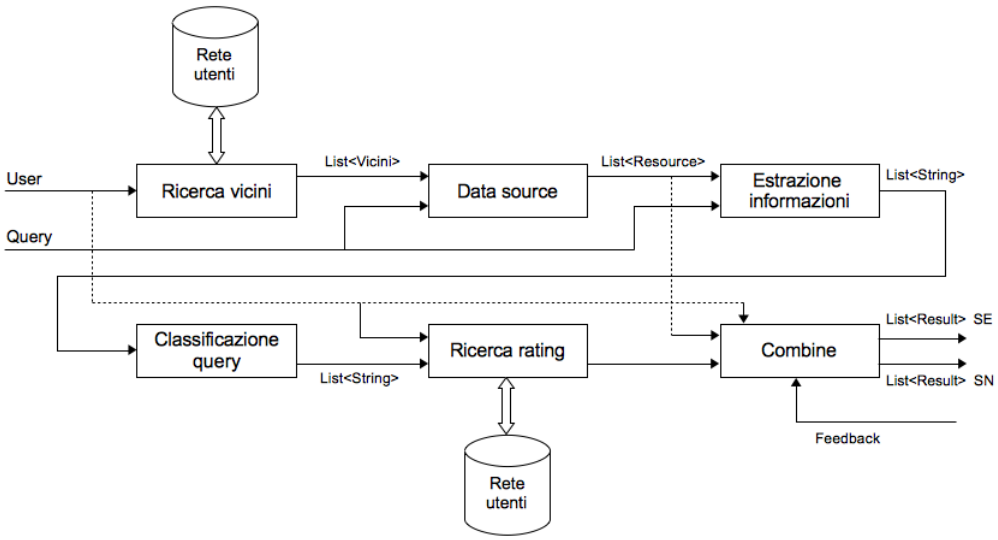

Figura 1.3.1: Architettura MilkRain

Vediamo invece un semplice utilizzo del sistema:

1. l'utente esegue una query
2. la query viene mandata al classificatore e classificata in topic
3. il sistema ottiene delle risorse grazie ai data source esterni (Google, Bing, Yahoo), le quali vengono classificate
4. le risorse collezionate e i rispettivi topic vengono mandati al modulo combine
5. il combine riordina la risorse in base alle valutazioni espresse dai vicini dell'utente e alla fiducia tra ques'ultimo e i suoi vicini
6. le risorse ordinate vengono presentate all'utente il quale può esprimere un giudizio da 1 a 5 per ognuna di queste
7. i valori nel database vengono aggiornati

1.3.1 Network

Uno degli aspetti fondamentali di MilkRain è la rete di utenti. Per questa ci sono tre elementi fondamentali:

1. l'utente
2. il collegamento tra due utenti
3. un topic generico

Utente

Un utente è collegato ad altri utenti e scambia dati con il sistema. I dati sono rappresentati dai giudizi delle risorse, le risorse stesse, il comportamento dell'utente e così via.
In generale, un utente MilkRain può essere un aggregatore di altri social network, ovvero un insieme di multiple identità appartenenti alla stessa persona (Facebook, Flickr, ecc); quindi è un contenitore di più "username" con le rispettive informazioni.

Link

Un link tra due utenti esprime due caratteristiche fondamentali: il topic condiviso tra gli utenti e il valore di trust tra essi. La presenza di un topic in un link implica che il primo utente si fida del secondo per un dato topic, ma il sistema può variare il valore di trust a seconda del comportamento del primo utente.
Un link tra due utenti è considerato unidirezionale.

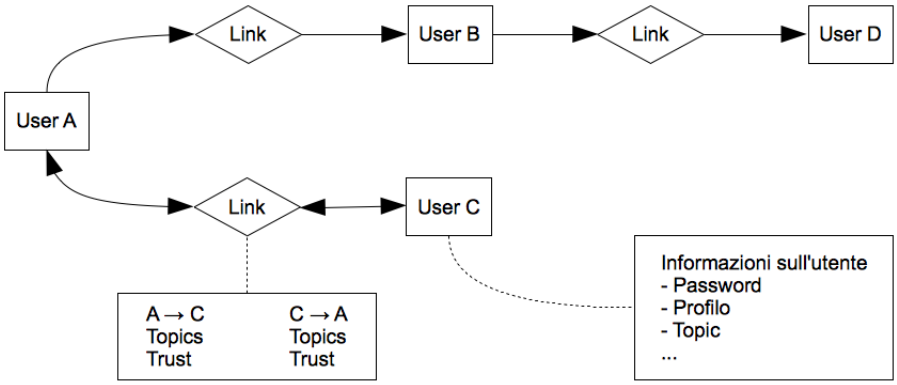

Figura 1.3.2: Un esempio di rete

1.3.2 Data Sources

Come data source si intende un' entità che accetta query e restituisce delle risorse. Non ci sono vincoli sul tipo di sorgenti ma la risposta deve essere controllata

dal sistema in quando deve essere trasformata in un certo modo.
I data source possono essere divisi in tipi; ci sono sorgenti che possono essere interrogate per ogni query e per questo appartengono alla categoria "Generic". Alcune sorgenti appartenenti a questa categoria sono Google, Bing e Yahoo.
Poi ci sono altri data source strettamente in relazione con il tipo di query (immagini, musica, ecc).

1.3.3 Classifier

Abbiamo detto che un link tra due utenti è rappresentato da un topic in comune, quindi è necessario classificare la query dell'utente in modo da capire se ci sono vicini che possono essere utili.

In generale non ci sono vincoli per il classificatore. Le informazioni utilizzate sono:

- la query
- la risposta dai data source
- la storia dell'utente

Il classificatore ha quindi il compito di prendere le risorse dai data source ed estrapolarne i topic.

1.3.4 Combine

Le risorse ottenute vengono filtrate, ordinate e modificate. Ciò che viene utilizzato per analizzare un risorsa è:

- il giudizio dato a questa da un vicino
- il valore di trust dell'utente nei confronti di un dato vicino
- la fiducia dell'utente nei confronti del sistema
- la storia dell'utente

1.3.5 Feedback

Questo modulo rappresenta il feedback dell'utente. Quest'ultimo deve avere l'opportunita di assegnare un giudizio alle risorse mostrate dal sistema; ogni voto viene memorizzato per il topic corrispondente alla query iniziale.
L'unico vincolo riguardante il tipo di giudizio è relativo al modulo combine, il quale ha l'informazione dalla sorgente dati (ad esempio il ranking originale) e dal feedback (ad esempio un feedback esplicito o click del mouse).

1.4 Lavoro svolto durante la tesi

I passi in più che sono stati realizzati con questa tesi sono:

- analizzare le difficoltà nella ricerca di un utente tramite i social network e i motori di ricerca
- affrontare il problema della sicurezza e della privacy in relazione al prelievo di questi dati dai vari social network per gli utenti e la sua soluzione tramite l'uso del protocollo di autenticazione OAuth
- l'aggiunta di un numero consistente di dati derivanti da identità online di altri siti quali sono i social network, nel nostro caso come elencato prima Facebook, Twitter, YouTube, Flickr, Digg, LinkedIn, Google+
- descrivere il database utilizzato e il modo in cui i dati vengono prelevati e successivamente memorizzati
- modifica del modulo combine
- conclusioni e possibili lavori futuri per la continuazione del progetto

Capitolo 2

Privacy e security

2.1 Ricerca dati utente tramite motore di ricerca

Proviamo a metterci nei panni di una persona qualsiasi che deve cercare sul web una persona generica e ricavare informazioni su quest'ultima, dalle più generiche alle più specifiche come ad esempio quali sono i suoi interessi o le interazioni con un altro utente (amico, un collega di lavoro, familiare, ecc.).
Se effettuo su Google una ricerca del mio nome e cognome i risultati che ottengo sono i seguenti:

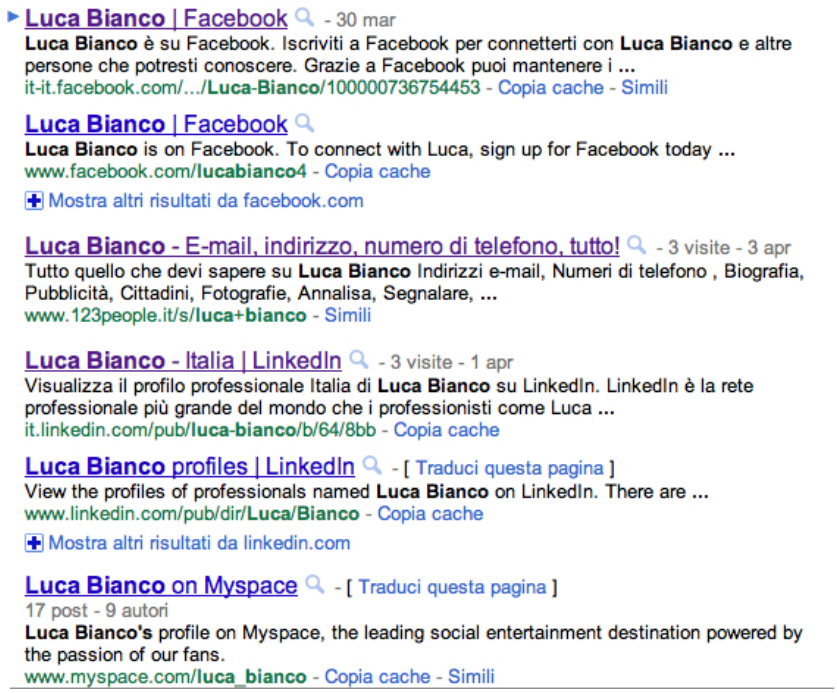

Figura 2.1.1: Esempio ricerca su Google

Come si vede dall'immagine i primi due risultati sono link al social network Facebook.com, successivamente troviamo 123people.it, linkedin.com e infine un altro social network, Myspace.com.

Se prendiamo in esame il caso di Facebook, anche solo per vedere l'identità della persona trovata devo essere registrato al social network, e anche supposto che lo sia devo nella maggior parte dei casi avere l'amicizia dell'utente per vedere le relazioni che intercorrono tra questa persona e un'altra, che sia un amico o un familiare.

Altro risultato della ricerca era un collegamento al sito Linkedin.com, il quale non è altro che un raccoglitore di informazioni lavorative e di cv degli utenti. Per vedere il profilo completo bisogna essere anche in questo caso registrati.
Rispetto a Facebook su Linkedin.com c'è meno interazione fra gli utenti e queste sono a livello lavorativo, mentre su facebook le relazioni sono basate più sul rapporto di amicizia.

Un altro sito che compariva tra i risultati della ricerca è 123people.it; questo è uno strumento gratuito di ricerca persone in tempo reale che controlla in ogni angolo del web. Si possono trovare informazioni centralizzate relativamente a persone provenienti da registri pubblici, numeri di telefono, indirizzi postali, immagini, video, indirizzi email. La ricerca comprende Facebook e altri siti di social networking come MySpace, LinkedIn, Xing, Wikipedia e molti alltri.

Un altro sito molto interessante è spokeo.com: inserendo nome e cognome di una persona restituisce una lista di tutte le persone trovate in internet con quel nome raccogliendo informazioni base(numero di telefono, abitazione,..) e profili dei social network. Anche questo è a pagamento se si vogliono vedere le informazioni più dettagliatamente.

Da quello che ho potuto capire è relativamente facile "trovare" una persona sul web, ma è cosa complessa o per meglio dire difficile, individuare quali sono le relazioni che intercorrono tra questa persona e altre, che queste siano amici, parenti, ecc..
Questo procedimento inoltre richiederebbe un lungo dispendio di tempo in quanto bisognerebbe controllare tanti aspetti della persona in questione, inoltre i dati a nostra disposizione sarebbero "grezzi". Per fare un esempio è difficile capire da qualche "Mi piace" su Facebook se effettivamente 2 utenti hanno interessi in comune oppure capire dato un item A che piace ad un utente X, se questo può piacere ad un altro utente Y in relazione con X.

Proviamo invece a immaginare di avere un software che aggrega tutte le diverse identità di un utente nel web in un'unica identità e raccoglie tutte le diverse interazioni di questa con gli altri utenti sparsi nella rete.
Risulterebbe davvero molto comodo per un qualsiasi utente effettuare la ricerca di questa persona e individuarla poi in internet. Oltre a rendere questo procedimento più semplice rispetto a una qualsiasi ricerca sul web, questo software mette a disposizione anche le diverse interazioni fra la persona cercata e tutta la

sua rete di conoscenze(amici, colleghi di lavoro, familiari,...) realizzando anche uno strumento di raccoglitore degli interessi di ogni utente e allo stesso tempo utile per consigliare ad utenti con simili caratteristiche ed interessi degli item.

Un aspetto "negativo" di questo software è che si incorre in problemi di security/privacy. Pensiamo appunto ad un software come quello descritto in precedenza che mette nelle mani di qualsiasi utente una lista di tutte le identità virtuali di una persona, delle sue abitudini (quali possono essere i vari stati sui Social Network), di tutte le relazioni che intercorrono tra questa e i suoi amici, immagini e cosa più importante tutti i suoi interessi e i suoi contatti.
Una lista delle cose che possono essere richieste da Facebook dopo una semplice iscrizione sono[1]:

Basic Subscriber Information:

- User Identification Number
- E-mail address
- Date and Time Stamp of account creation date displayed in Coordinated Universal Time
- Most Recent Logins (generally captures the last 2-3 days of logs prior to processing the request) in Coordinated Universal Time
- Registered Mobile Number

Expanded Subscriber Content:

- Profile Contact Information
- Mini-Feed
- Status Update History
- Shares
- Notes
- Wall Postings
- Friend Listing, with Friends Facebook IDs
- Groups Listing, with Facebook Group IDs
- Future and Past Events
- Video Listings, with filename

[1] http://www.techrepublic.com/blog/security/social-networking-sites-what-information-will-they-release-about-you/5009

Dove le foto utente possono includere foto caricate da altri utenti che hanno taggato la persona in questione.
Prendendo ancora l'esempio di Facebook, da ricerche effettuate[8] si ricavano questi interessanti dati

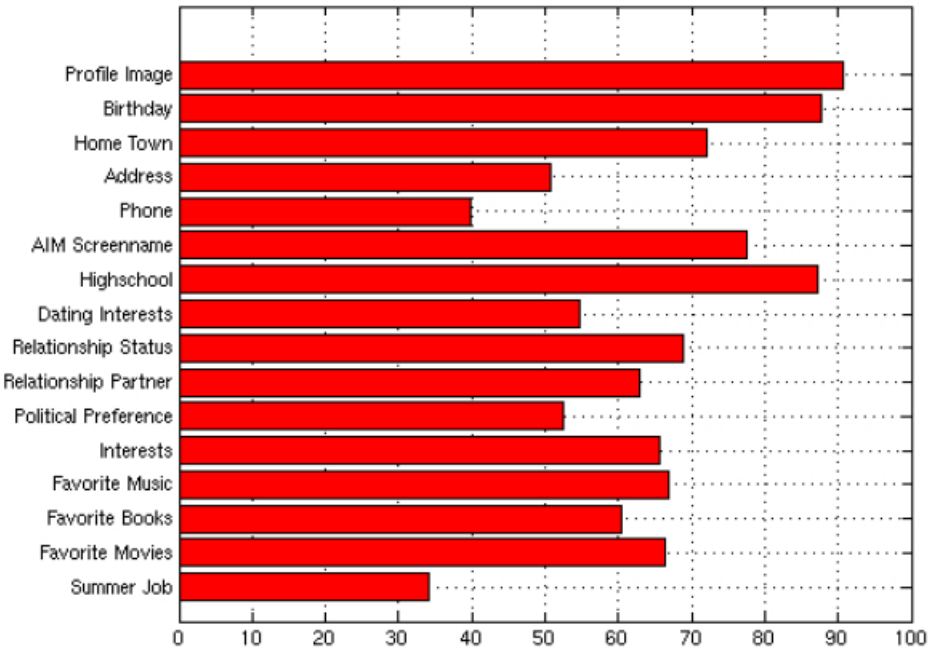

Figura 2.1.2: Informazioni ottenibili da un profilo Facebook

che rappresentano la percentuale di profili che rivelano vari tipi di informazioni personali.
Come si può vedere la gran parte degli utenti mette a disposizione molte informazioni riguardo se stesso. Pensiamo al nostro software in grado di raccogliere tutti questi dati e molti altri ancora più specifici, sarà semplice capire il perché sussiste un problema di sicurezza dei dati.

Quando si parla di Security è difficile non entrare nel discorso Privacy. La privacy coinvolge tutti gli aspetti che riguardano la protezione dei dati sensibili archiviati digitalmente ma in particolare è noto al grande pubblico con riferimento all'utilizzo di Internet[2]. Con il software che si vuole andare a realizzare in pochi semplici passi si andrebbe a mettere a disposizione di tutti una quantità considerevole di informazioni che potrebbero essere utilizzate dal più piccolo mal intenzionato all'hacker.

[2]http://it.wikipedia.org/wiki/Privacy#Privacy_e_Internet

Riporto quanto scritto sul blog di Stefano Bendandi riguardo al tema "Social network e privacy: quali sono i rischi per i dati personali?"[3]

> "..chiunque abbia accesso ad un qualsiasi servizio (l'accesso è libero, basta registrarsi, ed i controlli previsti in fase di registrazione sono spesso carenti) può acquisire e raccogliere i dati degli utenti, all'insaputa di questi ultimi, ed utilizzarli per le finalità più disparate (spam, pubblicità, attacchi diretti alla persona, discriminazione, ecc....) che, raramente, corrispondono a quelle per le quali è stato prestato il consenso.
> Questo inconveniente non può ritenersi superato nemmeno in virtù dei tanto sbandierati controlli di accesso, sia perchè a volte i dati personali, apparentemente non visibili, risultano accessibili mediante una semplice ricerca, sia per la debolezza intrinseca di molte impostazioni predefinite (pochi utenti si preoccupano di modificarle), sia perchè, in realtà, è molto facile diventare "amici" di tutti ed avere così accesso ai dati.
> Considerazioni analoghe le possiamo fare anche per i dati di natura, per così dire, secondaria (indirizzi ip, data e durata delle connessioni, profili visitati, messaggi scambiati, ecc...): questi sono di regola accessibili ai fornitori per asserite finalità miglioramento dei servizi ma, in realtà, non esistono di fatto adeguate garanzie di protezione e, soprattutto, di trattamento per le finalità consentite."

2.2 Ricerca dati utente tramite Social Network

Proviamo ora a fare un altro esperimento: vediamo cosa si riesce a costruire dell'identità di una persona avendo accesso ai più famosi Social Network e non come è stato fatto prima, cercando in un motore di ricerca. Nel fare questa analisi prenderemo in considerazione i seguenti siti:

- Facebook
- Twitter
- Linkedin
- YouTube

Prendiamo in esame Facebook e poniamo il caso io abbia creato il profilo di un utente non reale, solo per effettuare delle prove e delle analisi su di esso. Chiameremo questo utente John.
Come prima cosa per cercare di ricavare informazioni su questo utente fittizio devo essere registrato al Social Network in questione. Se John risulta appartenere alla mia rete di amici allora posso avere accesso a tutte le informazioni che lui ha reso disponibili sul suo profilo utente quali:

[3] http://stefanobendandi.blogspot.com/2008/04/social-network-e-privacy-quali-sono-i.html

- informazioni di base,
- informazioni di contatto (email e contatti ad altri social network),
- città dove vive e dove è nato,
- istruzione e lavoro,
- sport praticati,
- musica, libri, film, giochi, ...
- attività e interessi,
- foto e video,
- accesso alla bacheca dell'utente.

Nel caso in cui John non risulti essere mio amico i dati che potremmo essere in grado di raccogliere dipendono interamente da come è stata impostata la privacy per il suo profilo.
Facebook mette a disposizione degli utenti una vasta gamma di impostazioni della privacy che volendo rendono impossibile ad un qualsiasi utente di rintracciarci all'interno e all'esterno del social network.
Abbiamo prima visto che è possibile trovare un utente digitando il suo nome e cognome in motore di ricerca. Per alcuni il fatto di comparire nei risultati dei motori di ricerca rappresenta un modo per farsi conoscere e trovare, mentre molte persone non desiderano essere trovate.
Esiste per questo motivo una opzione che consente di impedire ai motori di ricerca di indicizzare il proprio profilo.

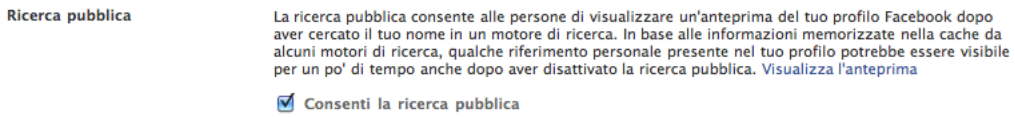

Figura 2.2.1: Selezione della ricerca pubblica

Deselezionando il flag "Consenti la ricerca pubblica" non sarà più possibile rintracciare il profilo Facebook dell'utente tramite un motore di ricerca.
Inoltre è anche possibile eliminare il proprio nome dalle ricerche effettuate utilizzando il motore di ricerca interno di Facebook. In questo modo il proprio nome viene rimosso definitivamente dai risultati delle ricerche.
Quindi ricapitolando solo effettuando questi due accorgimenti relativamente alle impostazioni della privacy, se io volessi cercare l'utente John creato in precedenza, sarebbe impossibile trovarlo perché non comparirebbe ne nei risultati dei motori di ricerca come Google, ne tra i risultati delle ricerche all'interno di Facebook. Non sarei quindi in grado di ricavare informazioni sull'utente.

Ci sono però altre impostazioni in merito alla privacy che è possibile personalizzare:

- è possibile rendere private le informazioni di contatto e mantenere privata la rete di amicizie

Figura 2.2.2: Privacy sulle informazioni di contatto

- modificare la privacy riguardo gli elementi che condivido

Figura 2.2.3: Privacy sugli elementi condivisi

- e le impostazioni riguardo agli elementi condivisi dagli altri

Figura 2.2.4: Privacy sugli elementi condivisi dagli amici

In conclusione possiamo dire che per quanto riguarda Facebook, trovare delle informazioni riguardo un utente che non è nella nostra rete di amicizie risulta essere una operazione molto complicata e risulta essere influenzata dal livello di privacy che questo utente ha impostato per il suo profilo.
Va detto che però la maggior parte degli utenti registrati su Facebook non modifica queste impostazioni e lascia quelle di default le quali permettono il rintracciamento della persona sui motori di ricerca interni ed esterni e lasciano pubbliche le informazioni base del profilo.

Consideriamo ora un altro Social Network come Twitter. Per quest'ultimo ricavare dati concernenti l'utente è molto più semplice poiché i profili sono pubblici e possono essere visti da chiunque. Bisogna segnalare che la quantità di informazioni a riguardo di un utente non sono molte in quanto Twitter è un Social Network in cui un utente può condividere un messaggio di 140 caratteri e una breve descrizione di se stesso; è anche possibile condividere nel messaggio un immagine e un video, che però fanno parte di quei dati grezzi, di cui parlavamo nel capitolo precedente, che risultano difficili da interpretare.
Come per Facebook c'è la possibilità di modificare le impostazioni riguardanti la privacy per far sì che gli utenti possano vedere il proprio profilo solo dopo che ne abbiamo accettato la richiesta.
Anche in questo caso va detto che sono veramente pochi gli utenti che scelgono questa via in merito alla privacy; inoltre anche una qualsiasi persona non registrata in Twitter può accedere ai profili degli utenti.

Simile a Twitter è Linkedin, infatti, anche per quest'ultimo i profili sono aperti, ma questa volta solo per utenti registrati al sito. Da questo possiamo ricavare una serie di informazioni sull'utente ricercato quali:

- informazioni personali,
- formazione,
- lingue conosciute,
- esperienza,
- specializzazioni,
- collegamenti con altri utenti.

Come per Facebook anche Linkedin mette a disposizione dell'utente un "pannello" per la gestione della privacy. Il numero di opzioni modificabili riguardo la privacy non raggiunge il numero di Facebook; va detto che quest'ultimo è però un Social Network molto diverso rispetto a quello in questione, dove gli utenti interagiscono con gli utenti in modo più diretto e rendono disponibili molte più informazioni rispetto a Linkedin nel quale le informazioni restano circoscritte all'ambito lavorativo/scolastico.
Le impostazioni modificabili sono:

- Attivare/disattivare la diffusione dell'attività
- Selezionare chi può vedere il feed dell'attività
- Selezionare le informazioni che gli altri vedono quando si visita il loro profilo

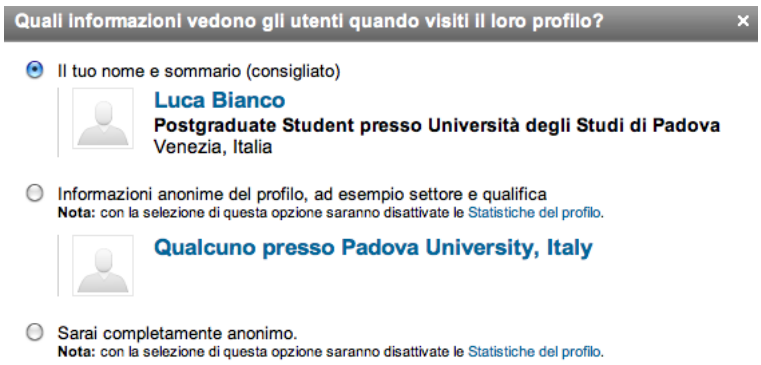

Figura 2.2.5: Privacy in LinkedIn

- Selezionare chi può vedere i tuoi collegamenti
- Modificare la foto del profilo e la visibilità
- Attivare/disattivare la condivisione dei dati con applicazioni di terze parti

Passiamo ora all'ultimo social network che prenderemo in esame ovvero YouTube. Per quest'ultimo pur conoscendo il nome della persona, questa risulta difficile da trovare in quanto nella maggior parte dei casi un utente crea un account con un username diverso dal nome reale.
Detto questo, ci metteremo nel caso in cui siamo a conoscenza del nome utente con cui la persona è registrata nel sito. Se accediamo al profilo di un utente le informazioni che possiamo trovare, oltre a quelle anagrafiche, sono il numero e la lista di iscritti al canale, sito web relativo all'utente, una breve descrizione e delle informazioni personali. Oltre a questi dati c'è naturalmente la lista di video caricati dall'utente e dei preferiti.
Anche in questo caso come per i dati multimediali in Facebook e Twitter, non è semplice da dei video ricavare informazioni riguardo una persona in quanto sono dati che vanno interpretati.

In conclusione com'era facile aspettarsi, la quantità maggiore di informazioni riguardo un utente è possibile ottenerla da Facebook, in quanto tra i social network citati è quello in cui gli utenti possono condividere la maggior parte delle informazioni personali, dai gusti musicali ai lavori praticati.
Come già scritto la mole di dati resi disponibili da un sito è proporzionale al numero di personalizzazioni possibili delle impostazioni sulla privacy. Questo è infatti un aspetto molto importante da prendere in considerazione quando si cerca di ottenere informazioni su una persona come abbiamo fatto noi; lo dimostra anche il fatto che in Facebook (su cui appunto si trovano la quantità più elevata di dati) dobbiamo prima ottenere l'amicizia dell'utente ricercato per "esplorare" il suo profilo a pieno.

Capitolo 3

Autenticazione OAuth

Prima di vedere come raccogliere le informazioni riguardo ad un utente nei rispettivi social network bisogna spendere del tempo nel spiegare un procedimento che precede la raccolta dati, ovvero l'autenticazione dell'utente tramite terzi (web application, siti web, etc.).
Le richieste vengono fatte dalla nostra applicazione tramite dei metodi API forniti dai siti web dai quali vogliamo prendere i dati sugli utenti. Per utilizzare questi metodi è necessario che gli utenti si autentichino al sito in questione con le loro credenziali. La cosa fondamentale di questo procedimento è che l'utente deve essere sicuro del fatto che le proprie credenziali non vengano memorizzate da terzi. L'autenticazione deve garantire ai service provider l'accesso da parte di terzi ai dati degli utenti, proteggendo contemporaneamente le loro credenziali.

Tutto questo viene realizzato tramite l'utilizzo di un protocollo di autenticazione chiamato OAuth[1].

3.1 Protocollo OAuth 1.0

OAuth è un protocollo aperto, sviluppato da Blaine Cook e Chris Messina[9] a partire dal novembre 2006. Questo protocollo permette l'autorizzazione di API di sicurezza con un metodo standard e semplice sia per applicazioni portatili che per pc fisso e web.

Cerchiamo di capire attraverso un semplice esempio il funzionamento del protocollo OAuth.
Molte auto di lusso vengono consegnate con una chiave supplementare per il parcheggiatore, che ne limita l'utilizzo. Queste chiavi dopo qualche chilomentro bloccano la macchina; alcune di queste anche altre parti come il bagagliaio.
L'idea alla base del protocollo OAuth è proprio questa: si da un accesso limitato alle risorse dell'utente tramite una chiave "secondaria", mentre ci si tiene una chiave "principale" per avere il controllo completo.

[1] http://it.wikipedia.org/wiki/OAuth

Prima di vedere un esempio reale, vediamo qual è la terminologia che vedremo nell'implementazione del protocollo OAuth:

- server: un server http in grado di accettare richieste OAuth autenticate. Inoltre è il sito o servizio web all'interno del quale le risorse protette sono memorizzate;
- client: un client http in grado di effettuare una richiesta OAuth;
- protected resource: risorsa ad accesso ristretto che può essere ottenuta dal server usando una richiesta OAuth autenticata;
- resource owner: un entità in grado di accedere e controllare risorse protette tramite l'utilizzo di credenziali autenticate con il server;
- credenziali: sono una coppia formata da un identificatore unico e un corrispondente segreto condiviso. OAuth definisce tre classi di credenziali: client, temporanee e token, usate rispettivamente per identificare e autenticare il client che effettua la richiesta, la richiesta di autorizzazione e la concessione di accesso;
- token: un identificatore univoco rilasciato dal server e usato dal client per associare le richieste autenticate con il proprietario della risorsa, la cui autorizzazione è richiesta o è stata ottenuta dal client. I token hanno una corrispondenza che è usata dal client per stabilire il possesso del token e "l'autorità" di rappresentare il possessore della risorsa.

3.2 Procedimento

John (resource owner) ha recentemente caricato alcune foto di una vacanza (protected resource) nel sito di photo sharing "photos.example.net" (server). A John piacerebbe usare il sito "printer.example.com" (client) per stampare una di queste foto. Solitamente il nostro utente accede a "photos.example.net" usando le sue credenziali: username e password.
Tuttavia, John non vuole condividere le sue credenziali con il sito web "printer.example.com" che necessita di accedere alle foto per stamparle. Al fine di fornire agli utenti un miglior servizio, "printer.example.com" si è fatto assegnare dal server delle credenziali:

```
Client Identifier:

    dpf43f3p214k3103

Client Shared-Secret:

    kdp4hf93k423kf44
```

Il sito "printer.example.com" ha anche configurato la sua applicazione in modo tale da usare gli endpoint listati nella documentazione API del sito "photos.example.net", che usa il metodo di firma "HMAC-SHA1":

```
Temporary Credential Request:
```

```
        https://photos.example.net/initiate

    Resource Owner Authorization URI:

        https://photos.example.net/authorize

    Token Request URI:

        https://photos.example.net/token
```

Prima che il client chieda a John di concedergli l'accesso alle foto, deve ottenere un insieme di credenziali temporanee da "photos.example.net" per identificare la richiesta di accesso. Per far questo il client manda la seguente richiesta HTTPS al server:

```
    POST /initiate HTTP/1.1

        Host: photos.example.net
        Authorization: OAuth realm="Photos",

            oauth_consumer_key="dpf43f3p2l4k3l03",
            oauth_signature_method="HMAC-SHA1",
            oauth_timestamp="137131200",
            oauth_nonce="wIjqoS",
            oauth_callback="http%3A%2F%2Fprinter.example.com%2Fready",
            oauth_signature="74KNZJeDHnMBp0EMJ9ZHt%2FXKycU%3D"
```

Il server convalida la richiesta e replica con un set di credenziali temporanee nel corpo della risposta HTTP (le interruzioni di linea sono solo per una migliore visualizzazione):

```
    HTTP/1.1 200 OK

        Content-Type: application/x-www-form-urlencoded
        oauth_token=hh5s93j4hdidpola&
        oauth_token_secret=hdhd0244k9j7ao03&
        oauth_callback_confirmed=true
```

Il client reindirizza John al Resource Owner Authorization endpoint del server per ottenere l'approvazione di John ad accedere alle sue foto private:

```
    https://photos.example.net/authorize?oauth_token=hh5s93j4hdidpola
```

Il server chiederà a John di accedere utilizzando la propria username e password, e se andata a buon fine, chiederà all'utente di consentire l'accesso alle sue foto. Approvata la richiesta John viene reindirizzato all'indirizzo di callback fornito dal client nella richiesta iniziale:

```
    http://printer.example.com/ready?
    oauth_token=hh5s93j4hdidpola&oauth_verifier=hfdp7dh39dks9884
```

La richiesta di callback informa il client che John ha completato il processo di autorizzazione. A questo punto il client richiederà un set di token usando le credenziali temporanee:

```
POST /token HTTP/1.1
Host: photos.example.net
Authorization: OAuth realm="Photos",
    oauth_consumer_key="dpf43f3p2l4k3l03",
    oauth_token="hh5s93j4hdidpola",
    oauth_signature_method="HMAC-SHA1",
    oauth_timestamp="137131201",
    oauth_nonce="walatlh",
    oauth_verifier="hfdp7dh39dks9884",
    oauth_signature="gKgrFCywp7rOOOXSjdot%2FIHF7IU%3D"
```

Il server convalida la richiesta e replica con i token nel corpo della risposta HTTP:

```
HTTP/1.1 200 OK
    Content-Type: application/x-www-form-urlencoded
    oauth_token=nnch734d00sl2jdk&oauth_token_secret=pfkkdhi9sl3r4s00
```

Dopo quest'ultimo step, il client è adesso pronto per effettuare la richiesta delle foto private:

```
GET /photos?file=vacation.jpg&size=original HTTP/1.1
Host: photos.example.net
Authorization: OAuth realm="Photos",
    oauth_consumer_key="dpf43f3p2l4k3l03",
    oauth_token="nnch734d00sl2jdk",
    oauth_signature_method="HMAC-SHA1",
    oauth_timestamp="137131202",
    oauth_nonce="chapoH",
    oauth_signature="MdpQcU8iPSUjWoN%2FUDMsK2sui9I%3D"
```

Il server "photos.example.net" convalida la richiesta e risponde con le foto richieste. Il client "printer.example.com" può continuare ad accedere alle foto private di John usando lo stesso insieme di token per l'intera durata della sessione o finché John revoca l'accesso.

3.3 Parametri

Cerchiamo ora di entrare un po' più nel dettaglio di quanto mostrato dal precedente esempio e cerchiamo di spiegare cosa sono tutti i parametri utilizzati.

I parametri che vengono utilizzati per costruire le URI[10] di richiesta precedentemente visti sono i seguenti:

- oauth_consumer_key: identificatore delle credenziali del client(si può dire che equivale allo username);
- oauth_timestamp: deve essere un valore intero. Se non diversamente specificato dalla documentazione del server, il timestamp è espresso nel numero di secondi dal 1 gennaio 1970, ore 00:00:00;
- oauth_nonce: è una stringa casuale, univocamente generata dal client per consentire al server di verificare che una richiesta non sia mai stata fatta prima e aiuta a prevenire attacchi quando la richiesta è fatta su un canale non sicuro. Il valore nonce deve essere univoco in tutte le richieste con stesso timestamp, credenziali client e combinazioni token;
- oauth_callback: un URI assoluto a cui il server reindirizzerà il proprietario della risorsa quando il passo di autorizzazione sarà completato.

Apriamo una finestra a parte per i restanti parametri: oauth_signature_method e oauth_signature.

Affinché il server possa verificare l'autenticità della richiesta e impedire accessi non autorizzati, il client deve dimostrare che è il legittimo proprietario delle credenziali. Questa operazione viene eseguita utilizzando la parte shared-secret di ogni set di credenziali.
OAuth fornisce tre metodi al client per dimostrare che è il corretto possessore delle credenziali: "HMAC-SHA1", "RSA-SHA1", e "PLAINTEXT". Questi sono generalmente indicati come metodi di firma anche se "PLAINTEXT" non comporta una firma. Inoltre "RSA-SHA1" utilizza una chiave RSA al posto delle comuni parti shared-secret associate alle credenziali del client.
Il client dichiara quale metodo viene utilizzato tramite il parametro oauth_signature_method. Si genera quindi una firma (o una stringa di valore equivalente) che verrà inserita nel parametro oauth_signature. Il server verificherà poi la firma. Il processo di firma non cambia la richiesta o i suoi parametri, con l'eccezione di oauth_signature. Nel nostro caso utilizzeremo sempre il metodo "HMAC-SHA1".

3.3.1 Stringa base

Per ottenere la firma come prima cosa bisogna costruire una stringa base. Questa altro non è che una concatenazione degli elementi delle richieste HTTP in una singola stringa. Quest'ultima è usata come input per i metodi visti sopra, escluso il "PLAINTEXT". Vengono dunque concatenati nell'ordine:

1. Il metodo di richiesta HTTP in maiuscolo. Per esempio "HEAD", "GET", "POST", ecc.
2. Un carattere "&".
3. L' URI della richiesta, dopo essere stata codificata.
   Ad esempio la richiesta HTTP:

```
GET /?q=1 HTTP/1.1
Host: www.example.net:8080
```

È rappresentata dalla sringa URI:

```
"https://www.example.net:8080/"
```

4. Un carattere "&".
5. I parametri di richiesta normalizzati, dopo essere stati codificati.

La codifica, effettuata ai punti 3 e 5, consiste nei seguenti passaggi:

- I valori di testo sono prima codificati come ottetti UTF-8[11] (Unicode Transformation Format, 8 bit) se non lo sono già. UTF-8 è una codifica dei caratteri Unicode in sequenze di lunghezza variabile di byte.
- I valori vengono preceduti dal carattere di escape usando il meccanismo della "percent-encoding" (%XX, dove le X rappresentano il valore esadecimale dell'ottetto) in questo modo:
  - I caratteri nell'insieme dei caratteri non riservati non devono essere codificati. Questi sono lettere maiuscole e minuscole, cifre decimali, "-", ".", "_", "~"
  - Tutti gli altri caratteri devono essere codificati
  - I due caratteri esadecimali usati per rappresentare i caratteri codificati devono essere in maiuscolo.

I parametri del punto 5 vengono normalizzati in stringhe nel seguente modo:

1. Prima si effettua la codifica di ogni parametro
2. I parametri sono ordinati per nome, usando un ordinamento per valore ascendente di byte. Se due o più parametri condividono lo stesso nome vengono ordinati per il loro valore
3. Il nome di ogni parametro è concatenato al suo corrispondente valore con un carattere "=" (codice ASCII 61) come separatore, anche se il valore è vuoto
4. Le coppie ordinate nome/valore sono concatenate insieme in una singola stringa utilizzando il carattere "&" (codice ASCII 38) come separatore.

Ad esempio, la seguente richiesta HTTP:

```
POST /request?b5=%3D%253D&a3=a&c%40=&a2=r%20b HTTP/1.1
    Host: example.com
    Content-Type: application/x-www-form-urlencoded
    Authorization: OAuth realm="Example",
```

```
    oauth_consumer_key="9djdj82h48djs9d2",
    oauth_token="kkk9d7dh3k39sjv7",
    oauth_signature_method="HMAC-SHA1",
    oauth_timestamp="137131201",
    oauth_nonce="7d8f3e4a",
    oauth_signature="djosJKDKJSD8743243%2Fjdk33klY%3D"
c2&a3=2+q
```

verrà normalizzata in questo modo:

Encoded	
Name	**Value**
oauth_consumer_key	9djdj82h48djs9d2
oauth_token	kkk9d7dh3k39sjv7
oauth_signature_method	HMAC-SHA1
oauth_timestamp	137131201
oauth_nonce	7d8f3e4a
c2	
a3	2%20q

Tabella 3.1: Codifica dei parametri

Sorted	
Name	**Value**
a3	2%20q
c2	
oauth_consumer_key	9djdj82h48djs9d2
oauth_nonce	7d8f3e4a
oauth_signature_method	HMAC-SHA1
oauth_timestamp	137131201
oauth_token	kkk9d7dh3k39sjv7

Tabella 3.2: Parametri ordinati per nome

Name=Value
a3=2%20q
c2=
oauth_consumer_key=9djdj82h48djs9d2
oauth_nonce=7d8f3e4a
oauth_signature_method=HMAC-SHA1
oauth_timestamp=137131201
oauth_token=kkk9d7dh3k39sjv7

Tabella 3.3: Concatenazione nome e valore

A questo punto vanno concatenate insieme in un'unica stringa:

```
a3=2%20q&a3=a&b5=%3D%253D&c%40=&c2=&oauth_consumer_key=9djdj82h48djs9d2&
oauth_nonce=7d8f3e4a&oauth_signature_method=HMAC-SHA1&
oauth_timestamp=137131201&oauth_token=kkk9d7dh3k39sjv7
```

3.3.2 HMAC-SHA1

Veniamo ora al passo finale, il metodo di firma HMAC-SHA1. L'algoritmo è implementato dalla maggior parte dei linguaggi di programmazione ed è definito come segue:

```
value = HMAC-SHA1(key,text)
```

Le variabili della funzione sono usate nel modo seguente:

text: insieme di valori della stringa base ottenuta come visto in precedenza
key: è la concatenazione dei valori di:

1. Il shared-secret del client, dopo essere stato codificato

2. Un carattere "&", che deve essere incluso anche quando entrambi i segreti sono vuoti

3. Il shared-secret del token, dopo essere stato codificato

value: è il valore del parametro "oauth_signature".

3.4 OAuth Work Flow

Proviamo a semplificare quanto esposto sopra tramite l'aiuto di un'immagine che mostra il flusso dello scambio di informazioni:

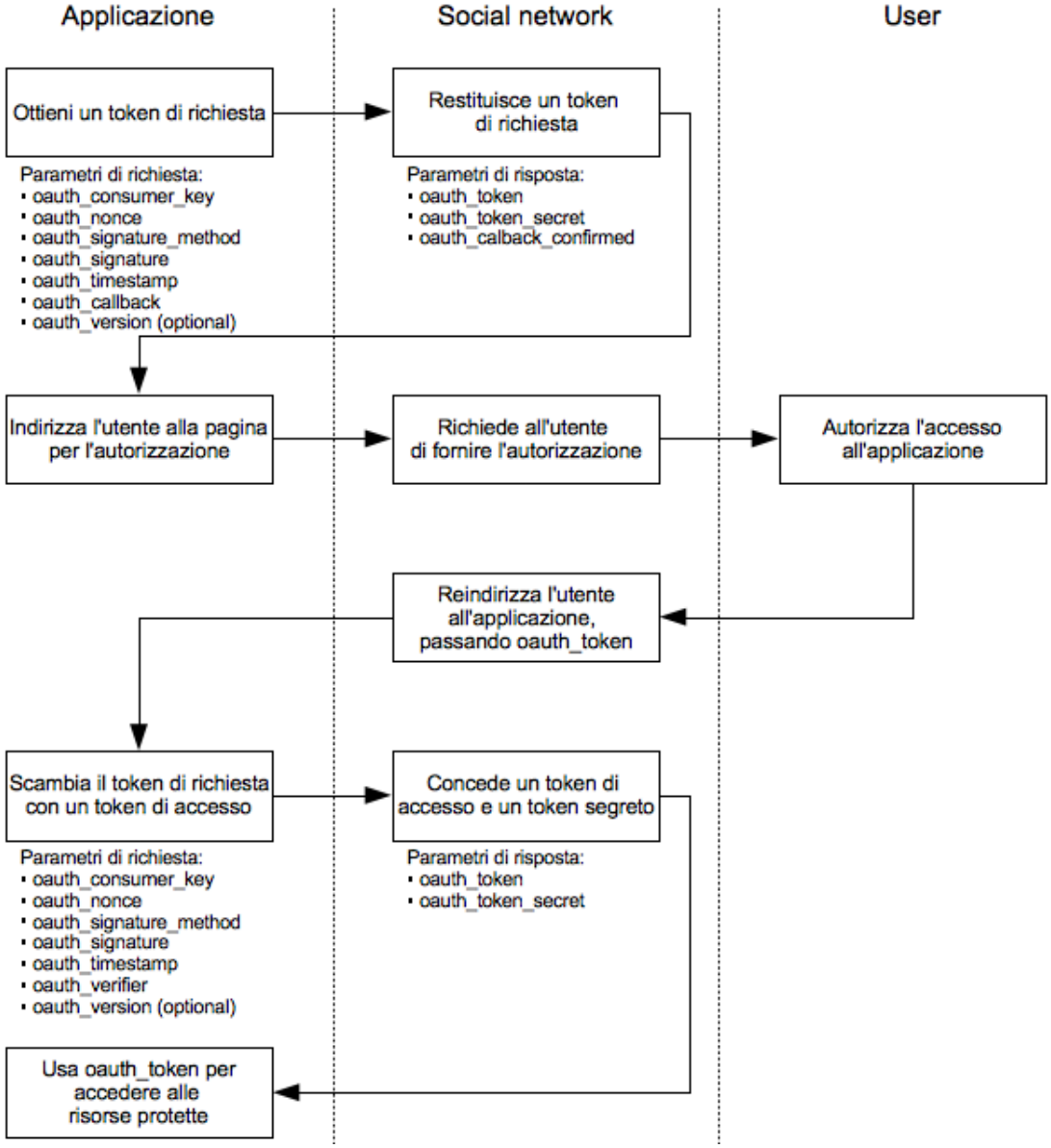

Figura 3.4.1: OAuth work flow

Il procedimento mostrato è identico per tutti i social network finora citati ad eccezione di Facebook che utilizza una versione avanzata del protocollo, ovvero OAuth 2.0.

3.5 Protocollo OAuth 2.0

Quest'ultima versione si basa sulle stesse fondamenta della precedente, ma risulta essere molto semplificata sotto molti aspetti tra cui il numero di

parametri utilizzati e il numero di passaggi per ottenere il token identificativo dell'utente.
Facciamo vedere un esempio pratico di una autenticazione su Facebook[2].
Dopo aver registrato la nostra applicazione ed aver quindi ottenuto un ID identificativo e una chiave segreta dell'applicazione possiamo iniziare il processo di autenticazione.
Come primo passo dobbiamo reindirizzare l'utente a una pagina di "dialogo" di facebook passando come parametri l'ID dell'applicazione e l'URL al quale il browser dell'utente sarà reindirizzato una volta che l'autorizzazione dell'applicazione sarà completata:

```
https://www.facebook.com/dialog/oauth?

    client_id=YOUR_APP_ID&redirect_uri=YOUR_URL
```

Se l'utente è già loggato, viene controllato il cookie di login memorizzato nel browser , autenticando l'utente. Se l'utente non risulta connesso, gli verrà chiesto di immettere le proprie credenziali:

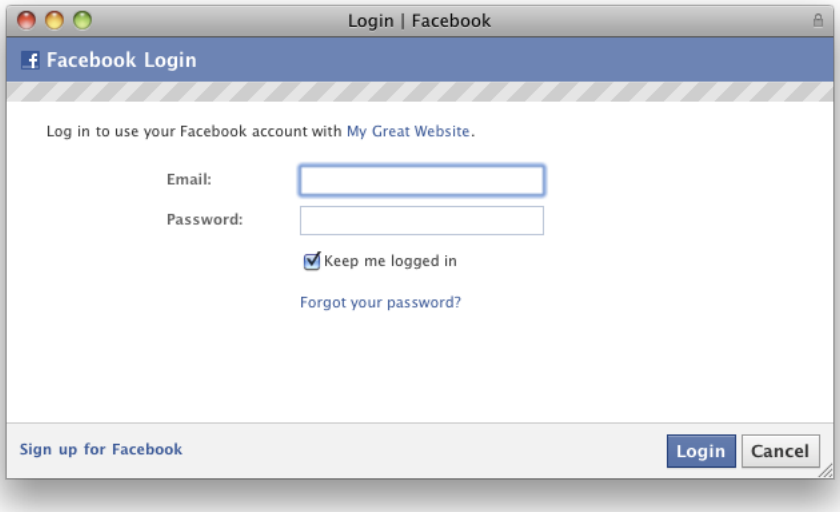

Figura 3.5.1: Facebook login

Una volta autenticato l'utente, verrà chiesto a quest'ultimo di autorizzare l'applicazione:

[2]http://developers.facebook.com/docs/authentication/

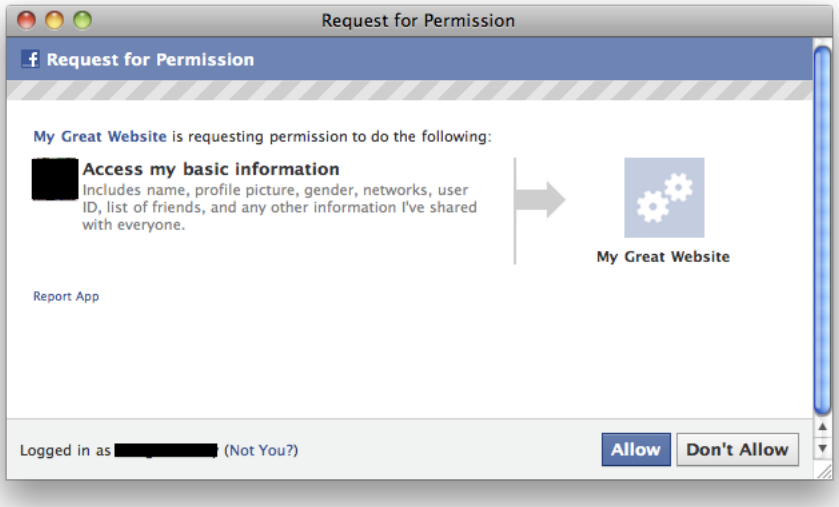

Figura 3.5.2: Esempio di richiesta di permesso per accedere ai dati dell'utente

Se l'utente preme "Don't Allow", l'applicazione non viene autorizzata. La finestra di dialogo reindirizzerà il browser dell'utente all'URL passato nel parametro redirct_uri con le seguenti informazioni d'errore:

```
http://YOUR_URL?error_reason=user_denied&

    error=access_denied&error_description=The+user+denied+your+request.
```

Se l'utente preme "Allow", l'applicazione risulta autorizzata. Anche questa volta il browser viene reindirizzato all'URL passato nel parametro redirct_uri, ma ora con un codice di autorizzazione:

```
http://YOUR_URL?code=A_CODE_GENERATED_BY_SERVER
```

Ora non ci resta che procedere verso l'ultimo passo che ci fornirà l'access token per effettuare le chiamate API. Per far questo è necessario passare il codice di autorizzazione appena ottenuto, la chiave segreta dell'applicazione, l'id utente el'URL di reindirizzamento all'endpoint https://graph.facebook.com/oauth/access_token. Otteniamo quindi il seguente link:

```
https://graph.facebook.com/oauth/access_token?

    client_id=YOUR_APP_ID&redirect_uri=YOUR_URL&
    client_secret=YOUR_APP_SECRET&code=THE_CODE_FROM_ABOVE
```

Se anche questo passo viene superato ci viene restituito l'access token con cui poter effettuare le chiamate API:

Figura 3.5.3: Esempio di risposta contente l'access token

Oltre all'access token ci viene restituito il parametro "expire" che rappresenta il numero di secondi entro i quali il token è valido. Una volta passato questo arco di tempo sarà necessario rifare i passaggi precedenti per ottenerne uno nuovo.

Capitolo 4

Database

4.1 Cassandra

Nel lavoro precedentemente iniziato da Benvenuti[1] si era deciso di implementare il database del sistema Milkrain utilizzando Cassandra.
Cassandra è una distribuzione open source interamente scritta in Java, è stata resa aperta da Facebook nel 2008 ed ora è un progetto dell'Apache Software Foundation utilizzato da molte compagnie come appunto Facebook, Twitter, Digg, Rackspace, Cloudkick, Cisco's WebEx e molte altre.
Apache Cassandra fa parte delle famiglie dei prodotti nosql, ovvero di quei software che nel processo di memorizzazione dei dati non utilizzano la sintassi SQL e il concetto di relazione. Risulta quindi essere un database distribuito creato per lavorare in cluster e per gestire una quantità elevata di dati.
Invece di utilizzare i classici concetti di tabella, tupla e relazione, Cassandra usa una modellizzazione dei dati column-oriented implementata tramite Hash e Array, memorizza cioè le informazioni in questa forma:

```
{ chiave: valore }
```

Affronteremo il modo in cui dati sono memorizzari nel database più avanti; cerchiamo ora di vedere il perchè la scelta dell'implementazione del database è ricaduta su Apache Cassandra.

4.2 Perchè Cassandra?

Le ragioni principali per la scelta di Cassandra sono le performance di velocità e scalabilità, le quali al momento non sono rilevanti ma lo saranno in una futura espansione con cluster costituiti da molti nodi. Basta infatti pensare al numero di utenti presenti anche nel solo Facebook e alla quantità di informazioni di cui è necessario tenere traccia per ognuno di essi e risulta subito evidente che la mole di dati sarà abbastanza considerevole.

Le caratteristiche di Cassandra sono:

1. Decentralizzazione: è un database distribuito su nodi identici e con consistenza dei cluster regolabile; significa che il server è installato in una configurazione clustered nella quale i nodi cooperano per ottimizzare e distribuire le informazioni. Non ci sono colli di bottiglia, né punti di rottura.

2. Elasticità: il throughput in lettura/scrittura aumenta linearmente con l'aggiunta di nuove macchine (nodi) al cluster senza downtime o disservizi alle applicazioni.

3. Fault Tolerance: i dati sono automaticamente duplicati su più nodi. Questo implica che in seguito a un eventuale crash non ci sia perdita di informazioni o lo stallo di Cassandra. I nodi in stallo possono essere tranquillamente rimpiazzati.

4. Durabilità: è un database pensato per applicazioni dove la possibile perdita di dati risulti essere critica, quando questo accade il sistema risolve il problema attraverso un procedimento di sincronizzazione dei dati.

Vediamo ora un'analisi delle performance[12] su un database MySql e un database Cassandra di dimensioni pari a 50Gb:

- MySql
  - 300 ms in scrittura
  - 350 ms in lettura
- Cassandra
  - 0.12 ms in scrittura
  - 15 ms in lettura

Come si può vedere Cassandra è inferiore (e quindi migliore), in termini di tempo, a MySql sia in scrittura che in lettura. Risulta quindi essere veramente conveniente se, come nel nostro caso, ci troviamo a dover "maneggiare" un importante numero di dati.

4.3 Data model

Abbiamo precedentemente detto che Cassandra utilizza un modello column-oriented; quest'ultimo è un po' più difficile da capire rispetto al modello chiave-valore.
In un modello chiave-valore si ha appunto la chiave che identifica univocamente un valore, il quale può essere strutturato o non strutturato. Quindi il modo migliore e più semplice per capire il modello column-oriented è quello di cominciare con il modello chiave-valore e immaginare che il valore sia una collezione di altri elementi del tipo chiave-valore.
In poche parole, questa è una specie di struttura dove HashMap sono incluse in altre HashMap.

Vediamo più nello specifico quali sono le strutture che si possono trovare in Cassandra[1].

4.3.1 Column

Sono il più basso livello di organizzazione dei dati. Questi elementi base contengono tuple composte di un timestamp, un nome e un valore. Il timestamp serve a Cassandra per capire qual'è il valore più recente e quindi per gestire i conflitti.
Le colonne vengono spesso rappresentate con la notazione JSON. Vediamone un esempio:

```
{
    ‘‘name’’ : ’’emailAddress’’,
    ‘‘value’’ : ’’foo@bar.com’’,
    ‘‘timestamp’’ : 123456789
}
```

4.3.2 SuperColumn

Le super colonne sono un passo più complesse; non sono altro che delle colonne nelle quali i rispettivi valori sono altre colonne. Si può dire in pratica che una super colonna è una specie di array ordinato di colonne.
Vediamo una immagine che aiuta a comprendere questa struttura:

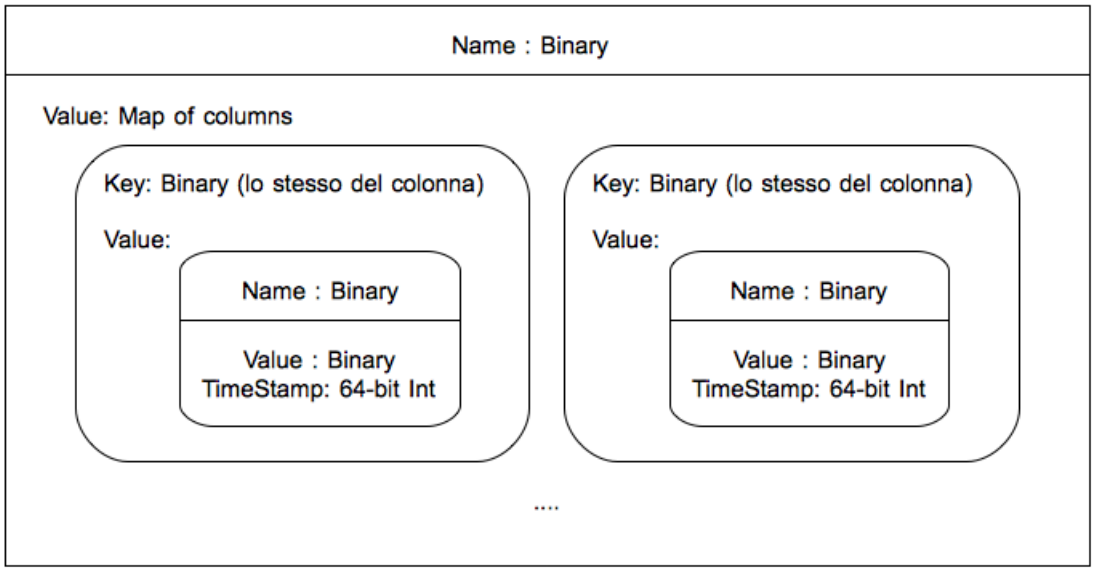

Figura 4.3.1: Schema raffigurante un super colonna

[1] http://wiki.apache.org/cassandra/DataModel

Come si può notare dalla figura non c'è limite al numero di colonne che una SuperColumn può contenere.

4.3.3 Column Family

Una ColumnFamily è un contenitore di colonne, l'analogo delle tabelle in un sistema relazionale. Una ColumnFamily mantiene la lista delle colonne ordinata, alle quali l'utente può far riferimento tramite il nome delle colonne.

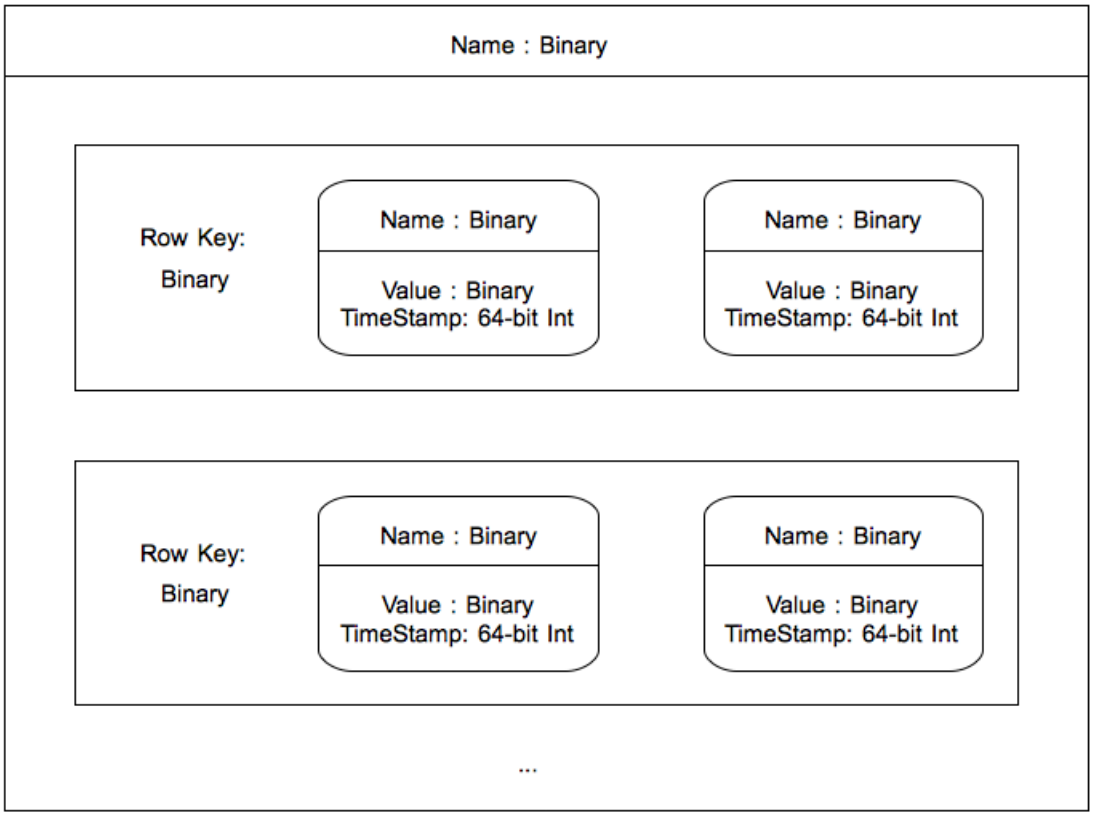

Figura 4.3.2: Esempio di column family

4.3.4 Row

In Cassandra ogni ColumnFamily è memorizzata in un file diverso e quest'ultimo è ordinato. Colonne correlate devono essere mantenute all'interno della stessa ColumnFamily.
La chiave di riga è ciò che determina in quale macchina i dati sono salvati. Così, per ogni chiave è possibile avere multiple ColumnFamilies associate ad essa.

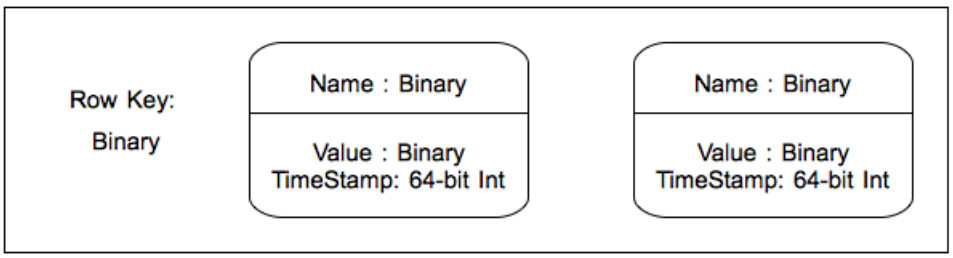

Figura 4.3.3: Esempio di riga

Una rappresentazione JSON della struttura key -> column families -> column è:

```
{
    "mccv":{
        "Users":{
            "emailAddress":{"name":"emailAddress", "value":"foo@bar.com"},
            "webSite":{"name":"webSite", "value":"http://bar.com"}
        },
        "Stats":{
            "visits":{"name":"visits", "value":"243"}
        }
    },
    "user2":{
        "Users":{
            "emailAddress":{"name":"emailAddress", "value":"user2@bar.com"},
            "twitter":{"name":"twitter", "value":"user2"}
        }
    }
}
```

Da notare che la chiave "mccv" identifica dei dati in due ColumnFamilies diverse, "Users" e "Stats". Questo non implica che i dati di queste colonne siano in relazione o in qualche modo collegati.

4.3.5 SuperColumn Family

Questa struttura non è altro che una mappa che contiene liste di SuperColumns e ad ognuna di queste fa riferimento una chiave di riga. Quindi l'unica differenza con le ColumnFamily è che dove queste hanno una collezzione di coppie nome/valore, le SuperColumn Family hanno sottocolonne.

4.3.6 KeySpace

Un keyspace è la prima dimensione di cassandra ed è il contenitore delle ColumnFamily. Esso è il punto di configurazione e gestione per le ColumnFamily, ed è anche la struttura sulla quale sono applicati gli inserimenti.

4.3.7 Cluster

Sono i nodi in una istanza di cassandra. I cluster possono contenere multipli keyspaces.

La tipica struttura a 4 dimensioni di Cassandra è quindi la seguente:

```
Keyspace
    -> Column Family
        -> Column Family Row
            -> Columns
              -> Valori
```

Nel caso in cui aggiungessimo delle SuperColumns:

```
Keyspace
    -> Super Column Family
        -> Super Column Family Row
            -> Super Columns
              -> Columns
                -> Valori
```

Capitolo 5

Modello utente

Dopo aver visto qual è la struttura del database sviluppato in Cassandra, vediamo in questo capitolo quale sarà la realizzazione vera e propria delle Column-Family per ogni sito web da noi preso in considerazione. In corsivo vengono riportati i parametri che risultano essere non sempre presenti.

5.1 Facebook

```
FACEBOOK_PROFILES
    Username
        Profile
            id =>
            username =>
            name =>
            first_name =>
            last_name =>
            birthday =>
            hometown =>
            gender =>
            bio =>
            location =>
            quotes =>
            political =>
            interested_in =>
            relationship_status =>
            relationship_with =>
            religion =>
            website =>
            link =>
        Favorite_teams
            id_team => name_team
                    :
        Favorite_athletes
```

```
            id_athlete => name_athlete
                    :
        Inspirational_people
            id => name
                    :
        Languages
            id_language => name_language
                    :
        Sports
            id_sport => name_sport
                    :
        ID_sport
            id_user => username
                    :
        Photo_tags
            id_photo => from_id
                    :
        Albums
            id_album => name_album
                    :
        Posts
            id_post => type_post
                    :
        Video_uploaded
            id_video => from_id
                    :
        Video_tags
            id_video => from_id
                    :
        Groups
            id_group => name_group
                    :
FACEBOOK_LIKES
    Username
        Category_1_name
            id => name
                :
        Category_2_name
            id => name
                :
FACEBOOK_GROUPS
    ID_group
```

```
        Information
            id =>
            name_owner =>
            id_owner =>
            name =>
            privacy =>
            icon =>
            link =>
            email =>
            description =>
FACEBOOK_PHOTOS
    ID_photo
        Picture
            name =>
            source =>
            height =>
            width =>
            link =>
            icon =>
            created_time =>
        From
            id =>
            name =>
        Tags
            id_user => name
                :
        Comments
            id_comment => from_id_user
                :
FACEBOOK_COMMENTS
    ID_comment
        Information
            from_id =>
            from_name =>
            message =>
            type => photo/album/post
FACEBOOK_ALBUMS
    ID_album
        Information
            from_id =>
            from_name =>
            name =>
            link =>
            cover_photo =>
```

```
            privacy =>
            numer_photos =>
            created_time =>
            updated_time =>
        Likes
            id_user => name
                :
        Comments
            id_comment => from_id_user
                :
FACEBOOK_VIDEOS
    ID_video
        Information
            from_id =>
            from_name =>
            name =>
            description =>
            picture =>
            source =>
            embed_html =>
        Tags
            id_user => name
                :
        Likes
            id_user => name
                :
        Comments
            id_comment => from_id_user
                :
FACEBOOK_WORKS
    Username
        ID_work
            id_employer =>
            name_employer =>
            location =>
            position =>
            description =>
            start_date =>
            end_date =>
            project => yes/no
            with => code(id_work,with)
        Code(ID_WORK,WITH)
            id_user => username
                :
```

Projects_username

ID_project

id =>
name =>
id_work =>
description =>
start_date =>
end_date =>
with => code(id_project,with)

Code(ID_PROJECT,WITH)

id_user => username
:

FACEBOOK_EDUCATIONS

Username

ID_school

school_name =>
type =>
classes => yes/no
year =>
degree =>
with => code(id_school,with)
concentration => code(id_school,concentration)

Code(ID_SCHOOL,WITH)

id_user => username
:

Code(ID_SCHOOL,CONCENTRATION)

id => name
:

Classes_Username

ID_class

id =>
name =>
id_school =>
description =>
with => code(id_class,with)

Code(ID_CLASS,WITH)

id_user => username
:

FACEBOOK_POSTS

ID_post

Information

from_id =>
from_name =>
type => link/video/status/photo

name =>
message =>
caption =>
link =>
description =>

To

id_user => name
:

Likes

id_user => name
:

Comments

id_comment => from_id_user
:

5.2 Twitter

TWITTER_PROFILES

Username

Profile

id =>
name =>
location =>
profile_image =>
url =>
description =>
friends_count =>

Followers

id => name
:

Friends

id => name
:

Searches

id => query
:

5.3 YouTube

YOUTUBE_PROFILES

Username

Profile

```
            firstname =>
            lastname =>
            username =>
            age =>
            about_me =>
            home_town =>
            books =>
            location =>
            movies =>
            music =>
            relationship =>
            school =>
            company =>
        Playlists
            id => title
                :
        Subscriptions
            1 => name
            2 => name
                :
        Contacts
            username => status
                :
        Favorites
            id_video => title
                :
        Uploads
            id_video => title
                :
YOUTUBE_PLAYLISTS
    ID_playlist
        Information
            title =>
            description =>
        Videos
            id => title
                :
YOUTUBE_VIDEOS
    ID_video
        Information
            title =>
            category =>
            duration =>
            view_count =>
            url =>
```

Tags

1 => name_tag
2 => name_tag
:

5.4 Flickr

FLICKR_PROFILES

Username

Profile

id =>
username =>
path_alias =>
real_name =>
location =>
photos_url =>
profile_url =>

Contacts

id_user => username
:

Favourites

id_photo => title_photo
:

Groups

id_group => name
:

Tags

id_photo => tag
:

Galleries

id_gallery => title_gallery
:

Photos

id_photo => title_photo
:

Sets

id_set => title_set
:

FLICKR_GALLERIES

ID_gallery

Information

```
            url =>
            title =>
            description =>
            count_photos =>
            owner_id =>
            owner_username =>
        Photos
            id_photo => title_photo
                :
```

5.5 LinkedIn

```
LINKEDIN_PROFILES
    Username
        Profile
            dateOfBirth =>
            firstName =>
            lastName =>
            headLine =>
            id =>
            imAccountName =>
            imAccountType =>
            industry =>
            interests =>
            locationCountry =>
            locationName =>
            mainAddress =>
            website =>
            phoneNumber =>
            publicProfileUrl =>
            specialities =>
            summary =>
            twitterAccount =>
        Groups
            id_group => name
                :
        Languages
            1 => italian
            2 => english
                :
        Connections
            id => name
                :
LINKEDIN_EDUCATIONS
    Username
```

ID_Education

schoolName =>
startDate =>
endDate =>
activities =>
degree =>
fieldOfStudy =>
notes =>

LINKEDIN_POSITIONS

Username

ID_Position

title =>
summary =>
startDate =>
endDate =>
isCurrent =>
companyID =>
companyIndustry =>
companyName =>
companyType =>

LINKEDIN_GROUPS

ID_Group

category =>
description =>
name =>
numMembers =>
shortDescription =>
siteGroupUrl =>

5.6 Digg

DIGG_PROFILES

Username

Profile

username =>
about =>
id =>
name =>
gender =>
diggs =>
comments =>
location =>
following =>
submissions =>
icon =>

Submissions

story_id => title

:

Diggs

story_id => title

:

SavedStories

story_id => title

:

Comments

comment_id => text

:

DIGG_STORIES

ID_Story

permalink =>
description =>
title =>
url =>
diggs =>
duggs =>
topic =>
comments =>

DIGG_COMMENTS

ID_Comment

text =>
diggs =>
down =>
up =>
story_id =>

5.7 Google+

GOOGLEPLUS_PROFILES

Username

Profile

id =>
displayName =>
description =>
gender =>
aboutMe =>
url =>
imageUrl =>

PlaceLived

```
            1 => Rome
            2 => Madrid
                :
        Activies
            id => type
                :
GOOGLEPLUS_ACTIVITIES
    ID_Activity
        Information
            type =>
            title =>
            url =>
            from_Name =>
            from_id =>
        Attachment
            type =>
            displayName =>
            content =>
            url =>
```

Capitolo 6

Social network come data source

Benvenuti nel suo elaborato definisce un Data Source come un'entità che accetta query e restituisce risorse. Non ci sono vincoli sul tipo di sorgenti dei dati ma la risposta deve essere prodotta o modificata in un certo modo dal sistema.
Attualmente le risorse sono ottenute facendo delle chiamate ai motori di ricerca Google, Yahoo e Bing. Il risultato consiste nell'URL, il titolo, il ranking e una breve descrizione della pagina.

Dopo aver prelevato i dati dai social network per gli utenti registrati al sistema Milkrain è evidente che entriamo in possesso di nuove risorse che entreranno a far parte dei data source. Quello che si vuole ora fare è:

- implementare una fase di ricerca all'interno del database oltre a quella attualmente in uso
- implementare un sistema di trust tra i vicini in Milkrain basato sulle informazioni ricavate dai social network
- ricavare un valore di "final ranking" per i risultati ottenuti dai social network

6.1 Ricerca

Prima di dedicarsi alla vera e propria ricerca bisogna capire quali sono i tipi di dati che possiamo ottenere sulla base della struttura del database.
Se diamo uno sguardo a come vengono memorizzati i dati all'interno di Cassandra si vede che le column family che contengo le risorse a noi necessarie sono:

- Flickr_Photos
- Facebook_Posts
- Digg_Stories

- GooglePlus_Activities
- YouTube_Videos

Per tutti questi dati sono stati salvati una descrizione, una URL, e un titolo. Risulta quindi abbastanza semplice l'utilizzo della classe "Resource" scritta da Benvenuti per "convertire" le nostre informazioni in risorse utilizzabili dal sistema milkrain.
Ciò che ancora non viene dato alle risorse è un valore di ranking, che per i risultati delle ricerche corrispondeva alla loro posizione nella lista restituita dai motori di ricerca. Discuteremo di questo punto nel prossimo paragrafo.

Concentriamoci ora sul come viene realizzata la ricerca vera e propria. Dato l'utente X, si andrà a cercare all'interno delle column family sopra elencate solo per i suoi vicini, supponiamo Y e Z.
Il metodo più semplice, data una query, è quello di andare a cercare all'interno dei parametri "title" e "description" della risorsa candidata le parole facenti parte della query e quindi effettuare un match tra queste.
Logicamente questa fase sarà preceduta da una di PreProcessing dove verranno eliminate tutte le "stop words" dai campi query, title e description. Fatto questo, si assumerà che un dato costituirà una risorsa se il match della query con i parametri supera una certa soglia, ottenuta tenendo in considerazione il numero di parole presenti nella descrizione.

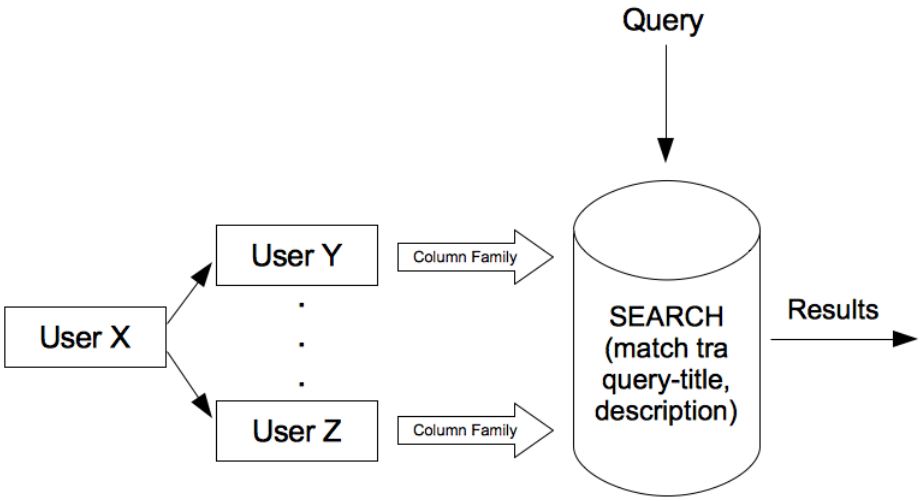

Figura 6.1.1: Flusso di operazioni per la ricerca

Una volta ottenuta la risorsa ciò che manca è un valore di ranking iniziale, che corrisponde alla posizione della risorsa nella lista dei risultati ottenuti dalla query. Questo valore viene assegnato facendo il rapporto tra il numero di match query-title/description e la lunghezza in termini di parole della query.

6.2 Analisi dati social network

Il sistema MilkRain è basato sulla fiducia di un utente nei confronti dei suoi vicini per un dato topic. A seconda del valore di questo parametro "trust" si darà maggior rilievo a risultati provenienti dal profilo di un vicino con valore trust elevato rispetto ad uno con valore basso, sempre in riferimento al topic in questione.

L'analisi dei profili dei social network dei vari utenti è stata fatta tenendo sempre a mente questo punto e quindi cercando di focalizzarsi su ottenere similarità tra un utente e i suoi vicini (similarità poi valutata in termini di topic).
Per far questo ci si è focalizzati su quelle parole che potevano essere di maggiore importanza rispetto ad altre, parole che chiameremo keywords e che vengono ottenute facendo delle analisi di frequenza.

Vediamo più nel particolare quali tabelle di Cassandra vengono utilizzate e come vengono analizzati i rispettivi dati.

1. **Facebook_Profiles**
   Per questa tabella si prendono in considerazione le informazioni relative ad atleti, squadre e sport preferiti dall'utente e il vicino in questione. Si confrontano i dati ricavati per vedere se presentano delle uguaglianze; il numero di dati uguali viene normalizzato secondo il numero totale di elementi e questo valore sarà assegnato al parametro trust tra utente e vicino per il topic. Quest'ultimo viene ricavato mandando in input al classificatore il tipo di preferito considerato inizialmente.
   Va sottolineato il fatto che il topic in comune viene inserito sole se il valore di trust è superiore a una soglia impostata allo 0,2.

2. **Facebook_Likes**
   Fanno parte di questa tabella tutte le pagine per le quali l'utente ha espresso un "Like". Le pagine vengono automaticamente suddivise da facebook in categorie, ed è proprio in base a questo attributo che sono state salvate in cassandra.
   In questo caso quindi i confronti verranno realizzati per categorie. Si prendono per l'utente la lista delle categorie(con rispettiva lista di pagine), per ognuna di queste si controlla sia presente tra quelle del vicino. Se il risultato è positivo si valuta quante sono le pagine in comune e infine se ne divide il valore per il numero totale di pagine per quella categoria.
   Come visto prima, se questo valore risulta superiore alla soglia di 0,2 la categoria verrà mandata in pasto al classificatore per ottenere dei topic da associare all'utente e al vicino con valore di trust pari al valore di similarità calcoltato.

3. **Facebook_Groups**
   In questo caso si prendono i gruppi in comune tra l'utente e il vicino e vengono trattati come delle risorse. Infatti per ogni gruppo nel database è salvata una descrizione, un nome e un link. Una volta creata una lista di risorse corrispondenti ai gruppi, questa viene mandata in input al ModifiedLingo creato da Benvenuti in modo tale da ottenere delle keywords,

appunto parole chiave che servono a rappresentare il contenuto delle risorse.
Queste parole vengono poi passate al classificatore per ottenere una lista di topic i quali saranno aggiunti ai topic comuni fra l'utente e il vicino con un valore di trust di default pari a 0,2.

4. **LinkedIn_Groups**
   L'analisi del contenuto di questa tabella è il medesimo di quello visto per i gruppi relativi a facebook.
   Bisogna tenere in considerazione che la "natura" dei gruppi di LinkedIn è di tipo lavorativo/scolastico e quindi i topic che verranno restituiti saranno differenti da quelli precedenti, proprio per questo è necessaria l'analisi anche di questa tabella.

5. **LinkedIn_Positions**
   Viene effettuata una analisi in frequenza di tutte le posizioni lavorative dell'utente e del vicino separatamente; si tengono solo le stringhe più ripetute e vengono quindi confrontate con quelle del vicino. Se si verifica una uguaglianza allora la stringa in questione viene inserita in una lista che verrà mandata in input al classificatore per ottenere il topic comune da inserire tra i due utenti.

6. **Digg_Stories**
   Per questa tabella ci viene in aiuto il fatto che per ogni storia memorizzata nel database è presente un attributo topic. Tutto quello che bisogna fare è prendere per l'utente e per ogni suo vicino la lista di topic, effettuare una analisi in frequenza per individuare quali sono quelli più ripetuti(anche in questo caso ogni topic verrà normalizzato per il numero totale di topic). Come visto in precedenza un topic si definisce ripetuto se il valore della frequenza normalizzato è superiore alla soglia di 0,2. A questo punto non resta che verificare che i topic frequenti siano in comune con quelli dei vicini(anche per questi viene fatto lo stesso procedimento).
   Infine ogni elemento della lista risultante viene classificato e inserito come topic comune tra i due utenti.

7. **YouTube_Videos**
   L'analisi è la medesima di quella vista per le storie di Digg solo che in questo caso al posto dei topic si prendono i tag dei video che altro non sono che parole utilizzate per descrivere i video.

8. **Flickr_Photos**
   Allo stesso modo di quanto fatto per i video di YouTube, anche in questo caso si prendono i tag utilizzati per descrivere le foto, su questi viene fatta l'analisi in frequenza e i risultanti vengono normalizzati per il numero totale di tag. La lista ottenuta viene quindi confrontata con quella dei vicini e se ci sono delle uguaglianze queste vengono mandate al classificatore e i topic risultanti inseriti a comune fra i due utenti.

6.3 Modifica blocco combine

Arrivati a questo punto, prima di poter mostrare all'utente milkrain il risultato della ricerca, occorre calcolare il valore chiamato "final ranking" che sta a rappresentare la posizione finale della risorsa nell'elenco dei risultati dopo aver valutato una serie di parametri.

Nella versione di Benvenuti questo valore veniva calcolato secondo il seguente algoritmo: per l'utente che ha eseguito la query vengono estratti i valori di trust per i ranking e per i rating di ogni topic restituito dal classificatore. Questi vengono chiamati coefficienti alfa e beta.
A questo punto ogni risorsa viene analizzata. Per ogni topic, tutti i vicini dell'utente che condividono lo stesso topic vengono estratti; se hanno votato la risorsa corrente per il topic in questione, i valori del voto e della fiducia nei confronti del vicino vengono prelevati dal database. Il rating finale è ottenuto come la media pesata dei giudizi, con i valori di trust come pesi:

$$final\_rating = \frac{\sum_{i=1}^{n_r} rating_i \times trust_i}{\sum_{i=1}^{n_r} trust_i} \tag{6.1}$$

dove n_r è il numero di voti per una data risorsa e un topic, $rating_i$ è il valore del voto e $trust_i$ è il corrispondente valore di fiducia.
Il ranking finale viene poi calcolato con la seguente:

$$final\_ranking = \alpha \times ranking + \beta \times final\_rating \tag{6.2}$$

dove alfa è la fiducia dell'utente nel valore di ranking per il topic in questione, beta è la fiducia dell'utente nel valore di rating per il topic dato, ranking è il valore ottenuto dai data source e final_rating è il risultato ottenuto dall'equazione precedente.

Si è deciso in questa prima fase di test, di separare i risultati della ricerca in due liste, quella comprendente i risultati dei motori di ricerca e quella con le risorse dei social network.
Quello che si vuole fare ora è assegnare a ogni risorsa ottenuta tramite la ricerca nei dati estrapolati dai social network un valore corrispondente al ranking finale. Questo valore viene calcolato in modo analogo da quanto fatto da Benvenuti per i risultati ottenuti dai motori di ricerca, in modo tale da mantenere una continuità con il lavoro prencedente.

La funzione per il calcolo del ranking finale sarà:

$$final\_ranking = \alpha_{SN} \times ranking + \beta_{SN} \times final\_rating \tag{6.3}$$

dove alfa rappresenta la fiducia dell'utente nel valore di ranking per il topic in questione in riferimento ai dati dei social network, beta è la fiducia dell'utente

nel valore di rating per il topic dato sempre in relazione ai social network, mentre final rating è calcolato in modo analogo a quanto fatto da Benvenuti, cioè:

$$final\_rating = \frac{\sum_{i=1}^{n_r} rating_i \times trust_{SN}}{\sum_{i=1}^{n_r} trust_{SN}} \tag{6.4}$$

con l'unica differenza che il valore di trust utilizzato è quello relativo alla fiducia di un utente nei confronti del suo vicino per un dato topic sulla base dei dati presi dai social network.

6.4 Feedback

Il meccanismo di feedback è realizzato mediante un rating esplicito da parte dell'utente per una risorsa basato su un sistema a stelle. Per ogni risultato di una ricerca l'utente può esprimere una valutazione da una a cinque stelle. Ogni stella viene mappata in una determinata posizione, secondo questo criterio:

- una stella: non rilevante o non pertinente, corrisponde alla trentunesima posizione
- due stelle: non abbastanza rilevante, corrisponde alla quindicesima posizione
- tre stelle: buono, corrisponde alla settima posizione
- quattro stelle: rilevante, corrisponde alla terza posizione
- cinque stelle: ottimo risultato, corrisponde alla prima posizione

I feedback vengono mappati in questo modo in quanto è stato scelto di limitare i risultati della ricerca fino a 20 risorse.

Vediamo ora l'algoritmo vero e proprio che utilizza le informazioni relative al giudizio delle risorse da parte degli utenti.
Il processo di feedback riceve in input la risorsa votata, il rating dell'utente, i topic relativi alla risorsa, l'utente e il rating generato dal modulo combine. Dal database vengono estratti i valori di ranking e rating dell'utente per i topic in questione (alfa e beta). A questo punto il valore del feedback viene confrontato con il rating e il ranking iniziale; se è più simile al ranking alfa viene incrementato di delta e il rating decrementato dello stesso valore, il contrario se il feedback è più simile al rating. Devono essere rispettati i seguenti vincoli:

- $\alpha_{SN} + \beta_{SN} = 1.0$
- $0.0 \leq \alpha_{SN}, \beta_{SN} \leq 1.0$
- $\Delta = 0.02$

I valori aggiornati di alfa e beta vengono memorizzati nel database.
Come passo finale vengono analizzati tutti i vicini che condividono con l'utente i topic relativi alla risorsa. Se quest'ultima è stata votata dal vicino vengono confrontati i due giudizi. Questi vengono considerati simili se identici o se differiscono di una stella; in questo caso il valore di trust dell'utente nei confronti del vicino viene incrementato di delta. Anche in questo caso vanno rispettati i seguenti vincoli:

- $0.0 \leq trust_{\mathrm{SN}} \leq 1.0$
- $\Delta_{\mathrm{trust}} = 0.02$

Capitolo 7

Risultati sperimentali

Questa ultima fase del progetto è servita per verificare il funzionamento del sistema sottoposto all'utilizzo di questo da parte di dodici utenti.
Ad ogni tester sono stati inviati due file, uno contenente le linee guida da rispettare nella fase di test e un altro contenente un listato di un centinaio di query da sottoporre al sistema.
Vediamo più precisamente cosa è stato richiesto ad ogni utente.

7.1 Testing

Il sistema è stato aperto in fase di test per un periodo di circa 2 settimane ed è stato testato da 8 utenti. Per ognuno di questi i prerequisiti necessari erano:

- essere registrato ad almeno uno tra i seguenti social network:
  - Facebook
  - Twitter
  - Flickr
  - YouTube
  - Google+
  - LinkedIn
  - Digg
- Avere almeno un vicino (amico in Milkrain) registrato ad almeno uno dei social network elencati al punto precedente.

Una volta registrato e aver aggiunto alla lista dei vicini almeno due utenti si esegue l'analisi dei dati importati dai social network, ottenendo così le similarità tra utente e vicino in fatto di topic.
A questo punto tutto ciò che l'utente deve fare è inserire nel campo "search" di Milkrain le parole presenti nella lista fornitagli e osservare i risultati ottenuti. Quest'ultimi sono la composizione di due ricerche: una effettutata tramite motori di ricerca (Google, Bing e Yahoo), l'altra fatta all'interno dei dati memorizzati

dai social network. Le due liste di risultati sono separate, prima compaiono i risultati ottenuti dai social network e poi quelle fornite dai motori di ricerca. Una volta effettuata al ricerca, l'utente deve valutare i risultati ottenuti dai social network. Valutare significa osservare i risultati, darne una valutazione (tramite le stelle presenti sotto ogni risultato) e se interessato cliccare sul link per visualizzare la pagina.

Le principali azioni di cui si tiene traccia durante questa fase sono:

- la registrazione di un nuovo utente
- l'aggiunta di un nuovo vicino
- la dichiarazione da parte dell'utente di un topic comune
- la classificazione di ogni query
- il final ranking di ogni risorsa
- il feedback di un utente
- il click su una risorsa

Nella tabella 7.2 sono riportati alcuni dati raccolti durante il periodo di test.

Numero totale di query	91
Numero totale di risorse	444
Numero di risorse mostrate	303
Numero di risorse votate	237
Numero di risorse cliccate	93
Numero di topic comuni	73

Tabella 7.1: Risultati generali

Una rappresentazione grafica della rete di utenti si può vedere in figura 7.1.1, dove il peso dei collegamenti tra gli utenti è proporzionale al numero di risorse condivise. Si noti che i link tra gli utenti sono unidirezionali.

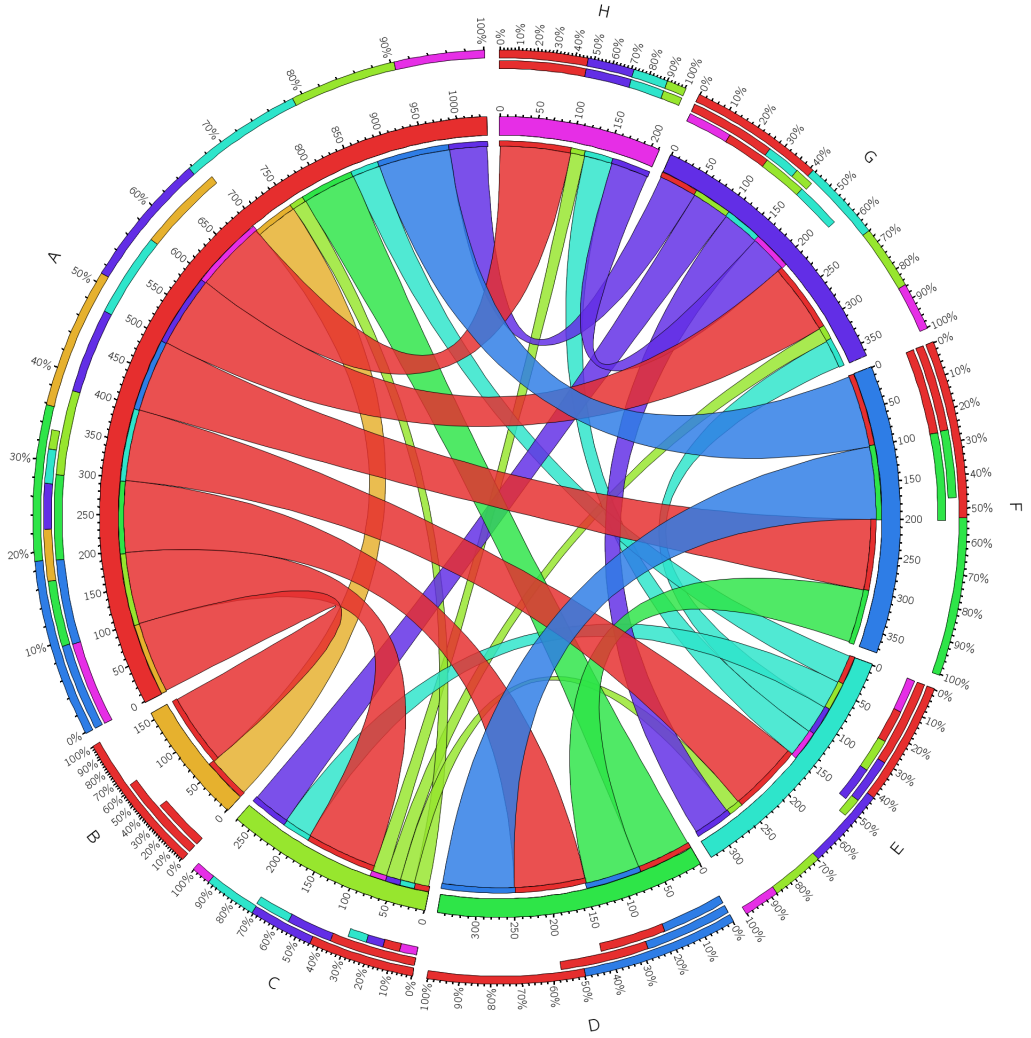

Figura 7.1.1: La rete degli utenti

La figura è stata ottenuta grazie al software "Circos Table Viewer"[1], strumento per la visualizzazione di tabelle.

7.2 Casi d'uso

7.2.1 Primo caso d'uso

Facciamo un primo esempio per mostrare le funzionalità del sistema.
Mostriamo come, anche se in fase di test, il feedback può rivelarsi uno strumento utile per ottenere risultati migliori quando la stessa query viene ripetuta più

[1] http://mkweb.bcgsc.ca/tableviewer/

volte.

Mettiamo nel caso in cui un primo utente, che chiameremo utente A, esegue la query “oasis” appartenente al topic arts/music. I risultati che vengono ritornati all’utente sono mostrati in figura 7.2.1

Figura 7.2.1: Query eseguita dall’utente A

A questo punto, l’utente A assegna un giudizio negativo alla prima risorsa (1 stella) e un giudizio positivo alla seconda (5 stelle).
Ora un secondo utente, chiamato utente B, che ha l’utente A come vicino e con cui condivide il topic arts/music, esegue la stessa query. Il risultato di questa seconda query è mostrato in figura 7.2.2. La risorsa che in precedenza ha ricevuto un giudizio positivo si trova ora al primo posto mentre, al contrario, la risorsa che ha avuto un giudizio negativo ha questa volta un ranking minore. L’utente B ha quindi avuto risultati migliori in relazione alla fiducia nel suo vicino per il topic arts/music.

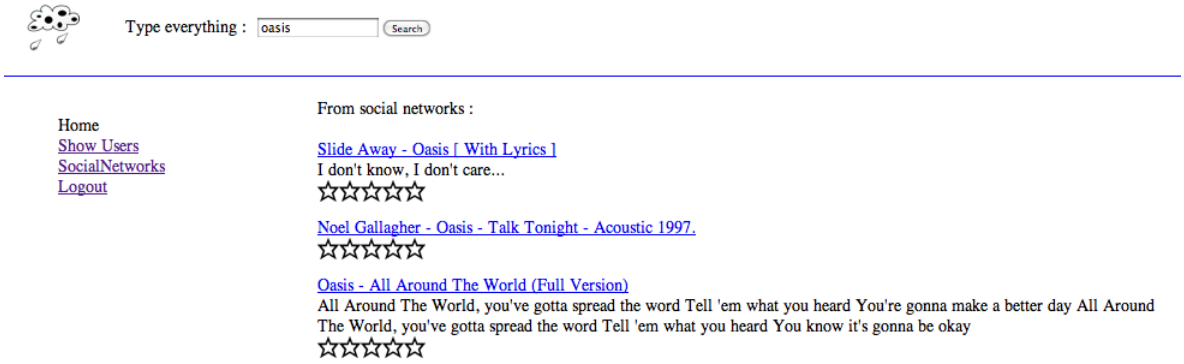

Figura 7.2.2: Query eseguita dall’utente B

	1 stella	2 stelle	3 stelle	4 stelle	5 stelle
Utente C	16,7	16,7	22,1	27,8	16,7
Utente D	11,8	11,8	23,5	23,5	29,4

Tabella 7.2: Distribuzione dei rating per gli utenti C e D

7.2.2 Secondo caso d'uso

Per questo esempio dimostrativo un insieme di query appartenenti allo stesso topic è stato assegnato ad un utente, che chiameremo utente C, il quale ha votato i risultati delle query. Lo stesso insieme di query è stato quindi assegnato ad un utente D, il quale è vicino di C e condivide lo stesso topic delle query.
Il risultato si può vedere nella tabella 7.2 dove sono evidenziati i rating delle risorse per i due utenti.

Si evidenzia per l'utente D un incremento della percentuale di risorse con voti di tre e cinque stelle e una diminuzione della percentuale di una e due stelle rispetto alle percentuali dell'utente C. Questo significa che l'utente D ha visualizzato migliori risultati rispetto a quelli del vicino.

Capitolo 8

Conclusioni

L'obiettivo che ci eravamo posti, e cioè implementare il progetto avviato da Benvenuti con le caratteristiche elencate nel primo capitolo, è stato raggiunto. Sono infatti stati aggiunti un numero considerevole di Data Sources provenienti dai più importanti social network implementando l'algoritmo OAuth per l'autenticazione degli utenti in modo tale da rispettare regole di privacy e security.

Si è scelto inoltre di mantenere una continuità con il lavoro precedente riguardo alla realizzazione dei moduli combine e feedback. Questo è stato fatto in modo tale da rendere più semplici espansioni future come ad esempio l'inglobamento in un'unica lista dei risultati ottenuti dai motori di ricerca e di quelli provenienti dai social network.

Il sistema Milkrain non risulta essere ad ogni modo ancora pronto per l'apertura al pubblico e ad un uso massiccio in quanto manca di alcune importanti caratterische come una opportuna interfaccia grafica. Servirebbe inoltre una graduale apertura pubblica in modo tale da popolare il database con un numero opportuno di risorse provenienti dai social network.

Osservando i risultati derivanti dalla fase di test possiamo ritenerci soddisfatti in quanto vi è un reale miglioramento di quelli che sono i risultati delle query effettuate dagli utenti dovuti ai feedback provenienti dalla rete di amicizie.

Eventuali lavori futuri riguardanti il sistema MilkRain possono essere:

- la possibilità di scegliere tra diverse tipologie di query (immagini, documenti, ecc)
- uno studio sulle performance, cercando di minimizzare i tempi di risposta
- il pieno supporto ad altre lingue; attualmente il classificatore è implementato in inglese
- la realizzazione di una opportuna interfaccia front end
- l'aggiunta di nuovi sistemi di feedback (impliciti ed espliciti)
- migliorare l'utilizzo dei dati forniti dagli utenti per i propri social network

- fondere i risultati dei social network con quelli dei motori di ricerca utilizzando un unico blocco combine

Appendice A

Dati utente

Al contrario di quanto fatto nel capito 5, prendiamo in analisi un social network alla volta e cerchiamo di approfondire a pieno quali sono i dati che questi mettono a disposizione riguardo gli utenti.
Lo schema generale sarà quello di far vedere cosa otteniamo facendo una chiamata a diverse API dei siti visti in precedenza e, in base a ciò che ci viene restituito, costruire il modello che utilizzeremo in Cassandra per memorizzare le informazioni sugli utenti.

A.1 Facebook

Per cominciare analizziamo Facebook[13], il social network che ci "consegna" la quantità più elevata di informazioni riguardo i suoi utenti.
Al centro di Facebook si trova il "grafo sociale" (social graph); le persone e le connessioni che queste hanno con tutto ciò che è di loro interesse. Le "Graph API" presentano una semplice e consistente vista del grafo sociale di Facebook, rappresentando uniformente gli oggetti(persone, foto, video, ecc.) e le connessioni tra di loro(relazioni di amicizia, contenuti condivisi, foto tags, ecc.).
Ogni oggetto nel grafo sociale ha un ID univoco. Si può accedere alle proprietà di un oggetto effettuando una richiesta di questo tipo: https://graph.facebook.com/ID.
Per esempio, la pagina ufficiale della piattaforma Facebook ha id 19292868552, quindi noi possiamo prendere le informazioni dell'oggetto facendo la richiesta seguente:

```
https://graph.facebook.com/19292868552
```

ottenendo come risposta l'oggetto JSON:

```
{
    "name": "Facebook Platform",
    "type": "page",
    "website": "http://developers.facebook.com",
    "username": "platform",
    "founded": "May 2007",
    "company_overview": "Facebook Platform enables anyone...",
```

```
    "mission": "To make the web more open and social.",
    "products": "Facebook Application Programming Interface...",
    "likes": 449921,
    "id": 19292868552,
    "category": "Technology"
}
```

Si può accedere a tutti gli oggetti allo stesso modo:

- Users: https://graph.facebook.com/btaylor (Bret Taylor)
- Pages: https://graph.facebook.com/cocacola (Coca-Cola page)
- Events: https://graph.facebook.com/251906384206 (Facebook Developer Garage Austin)
- Groups: https://graph.facebook.com/195466193802264 (Facebook Developers group)
- Applications: https://graph.facebook.com/2439131959 (the Graffiti app)
- Status messages: https://graph.facebook.com/367501354973 (A status message from Bret)
- Photos: https://graph.facebook.com/98423808305 (A photo from the Coca-Cola page)
- Photo albums: https://graph.facebook.com/99394368305 (Coca-Cola's wall photos)
- Profile pictures: https://graph.facebook.com/bianco.luca/picture (your profile picture)
- Videos: https://graph.facebook.com/817129783203 (A Facebook tech talk on Graph API)
- Notes: https://graph.facebook.com/122788341354 (Note announcing Facebook for iPhone 3.0)

Tutti gli oggetti nel grafo sociale di Facebook sono connessi agli altri tramite relazioni. Ad esempio Bret Taylor è un fan della pagina della Coca-Cola, oppure è un amico di Arjun Banker. Queste nelle API di Facebook vengono chiamate connessioni. Si possono esaminare le connessioni tra gli oggetti usando la struttura URL https://graph.facebook.com/ID_OBJECT/CONNECTION_TYPE.
La sezione per gli sviluppatori del social network in questione ci fornisce già una serie di tabelle che illustrano i parametri che vengono restituiti per ogni interrogazione alle API. Vediamo di seguito un elenco di queste interrogazioni con le rispettive tabelle (con elenco delle connessioni).

Utente		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/me	Restituisce i dati relativi all'utente	ID, name, first_name, middle_name, last_name, gender, languages, profile_link, username, bio, birthday, education, email, hometown, interested_in, location, political, favorite_athletes, favorite_teams, quotes, relationship_status, religion, website, work.
Connessione	**Descrizione**	**Risultato**
accounts	Le applicazioni e le pagine possedute dall'utente	Array di oggetti contenti ID, category, name
activities	Le attività elencate nel profilo utente	Array di oggetti contenenti activity id, name, category, create_time
albums	Gli album creati dall'utente	Array di oggetti album
books	I libri elencati nel profilo utente	Array di oggetti contenenti book id, name, category, create_time
family	Le relazioni familiari dell'utente	Array di oggetti contenenti id, name, relationship
friendlists	Le liste di amicizie dell'utente	Array di oggetti contenenti id, name
friend	Gli amici dell'utente	Array di oggetti contenenti id e name
games	Giochi che l'utente ha aggiunto alla sezione "Arte e intrattenimento" del suo profilo	Array di oggetti contenenti id, name, category, created_time
groups	I gruppi ai quali l'utente appartiene	Array di oggetti contenenti version, name, id, administrator (se l'utente lo è)
interests	Gli interessi presenti nel profilo utente	Array di oggetti contenenti id, name, category, created_time
likes	Tutte le pagine che piacciono all'utente	Array di oggetti contenenti id, name, category, created_time
movies	I film elencati nel profilo	Array di oggetti contenenti id, name, category, created_time
music	La musica elencata nel profilo	Array di oggetti contenenti id, name, category, created_time
photos	Le foto in cui l'utente è taggato	Array di oggetti Photo
post	I post dell'utente	Array di oggetti Post
videos	I video in cui l'utente è taggatto	Array di oggetti Video

Tabella A.2: Risorse disponibili per l'utente

Album		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_album	Restituisce i dati relativi ad un album	Id, from, name, description, location, link, cover_photo, privacy, count, type, created_time, updated_time
Connessione	**Descrizione**	**Risultato**
photos	Le foto contenute nell'album	Array di oggetti Photo
likes	I like fatti a questo album	Array di oggetti contenenti id, name, message, create_time
comments	I commenti fatti all'album	Array di oggetti contenenti id, name

Tabella A.4: Informazioni per l'oggetto album

Comment		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_comment	Restituisce i dati relativi ad un commento	Id, from, message, created_time, likes, user_likes, type
Connessione	**Descrizione**	**Risultato**
likes	I like fatti a questo album	Array di oggetti contenenti id, name dell'utente a cui piace il commento

Tabella A.6: Informazioni per l'oggetto comment

Group		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_group	Restituisce i dati relativi ad un gruppo	Id, version, owner, name, description, link, privacy
Connessione	**Descrizione**	**Risultato**
members	Tutti gli utenti che fanno parte del gruppo (al momento può restituire solo i primi 500)	Array di oggetti contenenti id, name.

Tabella A.8: Informazioni per l'oggetto group

Link		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_link	Restituisce le informazioni su un link condiviso su una bacheca	Id, from, link, name, comments, description, message, picture, created_time, type
Connessione	**Descrizione**	**Risultato**
comments	I commenti fatti al link	Array di oggetti contenenti id, from, message, created_time.
likes	I like fatti a questo link	Array di oggetti contenenti id, name dell'utente a cui piace il link

Tabella A.10: Informazioni per l'oggetto link

Photo		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_photo	Restituisce le informazioni su una foto	Id, from, tags, name, picture, source, height, width, images, link, created_time, updated_time, position
Connessione	**Descrizione**	**Risultato**
comments	Tutti i commenti alla foto	Array di oggetti contenenti id, from, message, created_time.
likes	Utenti a cui piace la foto	Array di oggetti contenenti id, name
tags	Gli utenti taggati nella foto	Tag con name e ID

Tabella A.12: Informazioni per l'oggetto photo

Post		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_post	Effettuando questa richiesta ci vengono restituiti oltre ai post anche i messaggi di stato.	Id, from, to, message, picture, link, name, caption, description, source, properties, icon, type, likes, comments, object_id, application, created_time, updated_time
Connessione	**Descrizione**	**Risultato**
comments	Tutti i commenti al post	Array di oggetti contenenti id, from, message, created_time.
likes	I like al post	Array di oggetti contenenti id, name

Tabella A.14: Informazioni per l'oggetto post

Video		
Risorsa	**Descrizione**	**Parametri**
graph.facebook.com/ ID_video	Restituisce le informazioni su un video	Id, from, tags, name, description, picture, embed_html, source, created_time, updated_time, comments
Connessione	**Descrizione**	**Risultato**
comments	Tutti i commenti al post	Array di oggetti contenenti id, from, message, created_time.
likes	I like al post	Array di oggetti contenenti id, name

Tabella A.16: Informazioni per l'oggetto video

A.2 Twitter

Twitter fornisce dei metodi API[14] per quasi tutte le funzionalità che si possono trovare nel sito web. Anche in questo caso, come per Facebook, ogni oggetto è identificato da un codice univoco (sia l'oggetto un utente, un tweet, ecc.).
Nella documentazione delle API che ci viene fornita c'è un indicatore della versione relativa ai metodi e che va segnalata ogni volta che facciamo una richiesta al server. Questo significa che tutte le richieste dei metodi API saranno di questo tipo:

```
https://api.twitter.com/1/friends/ids.json
```

In questo esempio chiediamo al server gli id di tutti gli amici dell'utente attualmente loggato in Twitter. La risposta ci verrà restituita in formato JSON come da noi richiesto:

```
{
    "previous_cursor": 0,
    "ids": [
        143206502,
        143201767,
        777925
    ],
    "previous_cursor_str": "0",
    "next_cursor": 0,
    "next_cursor_str": "0"
}
```

Vediamo ora un elenco di quelli che sono i metodi più importanti e più utili al fine di arricchire il nostro database con le informazioni sugli utenti.

Friends & Followers		
Risorsa	**Descrizione**	**Parametri**
followers/ids	Restituisce un array di ID numerici per ogni utente che sta seguendo l'utente.	ID_follower
friends/ids	Restituisce un array di ID numerici per ogni utente che l'utente segue.	ID_friend

Tabella A.18: Informazioni per friends e followers

Users		
Risorsa	**Descrizione**	**Parametri**
users/lookup	Restituisce informazioni fino ad un massimo di 100 di utenti specificati per ID o screen_name.	Array di users/show
users/show	Ritorna dati sull'utente autenticato.	name, profile_image_url, location, ID, url, followers_count, description, time_zone, friends_count, screen_name.

Tabella A.20: Informazioni per ogni utente

Favorites		
Risorsa	**Descrizione**	**Parametri**
favorites	Restituisce i 20 più recenti status preferiti dall'utente autenticato.	ID, text, retweeted, geo, in_reply_to_user_ID, array di user/show

Tabella A.22: Lista dei dati per i preferiti

Saved Searches		
Risorsa	**Descrizione**	**Parametri**
saved_searches	Restituisce le ricerche salvate dall'utente autenticato.	ID, query, name, created_at

Tabella A.24: Lista delle informazioni sulle ricerche salvate

Timelines		
Risorsa	**Descrizione**	**Parametri**
statuses/ user_timeline	Ritorna i 20 status più recenti postati dall'utente autenticato.	Per ogni status: ID, text, retweeted, geo, in_reply_to_user_ID, array di user/show

Tabella A.26: Informazioni sulla timeline

A.3 YouTube

YouTube mette a disposizione degli sviluppatori lo strumento "Data API[15]" che permette di far autenticare l'utente e di ottenere informazioni su di esso come playlist, video caricati, preferi, amici, ecc.
Vediamo di seguito la struttura di un esempio di una richieta alle API di YouTube che ci permette di ottenere le informazioni base del profilo utente:

```
https://gdata.youtube.com/feeds/api/users/default
```

dove "default" indica l'utente attualmente autenticato. Le risposte alle interrogazioni tramite le API, come per tutti gli altri social network, sono in formato JSON.
Vediamo ora più nello specifico quali sono i metodi che abbiamo utilizzato per costruire il profilo dell'utente in cassandra.

Video		
Risorsa	**Descrizione**	**Parametri**
gdata.youtube.com/feeds/api/videos/videoid	Restituisce i dati relativi al video richiesto	title, category, duration, view_count, url, description, list_tags

Tabella A.30: Informazioni sui video

Users		
Risorsa	**Descrizione**	**Parametri**
gdata.youtube.com/feeds/api/users/default	Restituisce il profilo dell'utente attualmente collegato.	firstName, lastName, aboutMe, age, username, books, gender, company, hobbies, hometown, location, movies, music, occupation, school
gdata.youtube.com/feeds/api/users/default/contacts	Restituisce la lista dei contatti dell'utente	username, status
gdata.youtube.com/feeds/api/users/default/playlists	Restituisce la lista di tutte le playlist dell'utente	ID, title, summary, description, videos_id
gdata.youtube.com/feeds/api/users/default/subscriptions	Restituisce la lista dei canali a cui l'utente è iscritto	list_name_subscription
gdata.youtube.com/feeds/api/users/default/uploads	Restituisce la lista dei video caricati dall'utente	ID, title, description

Tabella A.28: Informazioni sull'utente YouTube

A.4 Flickr

Con oltre 5 miliardi di foto (molte contenenti metadati come tag e posizioni geografiche), la community di Flickr produce un'incredibile varietà di dati. L'API di Flickr[16] rappresenta la modalità di accesso a quei dati. Quasi tutte le funzionalità eseguite da flickr.com sono disponibili tramite l'API.
Ancora in questo caso, come era logico aspettarsi, l'API Flickr utilizza identificatori univoci per utenti, foto, set di foto e altri oggetti.
Le richieste da effettuare tramite HTTP hanno una struttura simile a quelle viste precedentemente:

```
http://api.flickr.com/services/rest/?method=flickr.contacts.getList
```

Da questa chiamata si ottiene la lista dei contatti dell'utente che ha effettuato l'accesso al sito, ad esempio:

```
{
```

```
    "contacts":
    {
        "page": 1,
        "pages": 1,
        "per_page": "1000",
        "perpage": "1000",
        "total": 4,
        "contact":
        [
            {
                "nsid": "12037949629@N01",
                "username": "Eric",
                "iconserver": 0,
                "iconfarm": 0,
                "ignored": 0,
                "realname": "Eric Costello",
                "friend": 1,
                "family": 0,
                "path_alias": "",
                "location": ""
            },
            {
                "nsid": "12037949631@N01",
                "username": "neb",
                "iconserver": 0,
                "iconfarm": 0,
                "ignored": 0,
                "realname": "Cal Henderson",
                "friend": 1,
                "family": 0,
                "path_alias": "",
                "location": ""
            }
        ]
    },
    "stat": "ok"
}
```


Vediamo anche in questo caso una lista di quelli che sono i metodi che offre Flickr per interagire col sito e quindi per raccogliere tutti i dati di cui abbiamo bisogno.

Contacts		
Risorsa	**Descrizione**	**Parametri**
flickr.contacts.getList	Restituisce la lista dei contatti dell'utente.	ID, username, realname

Tabella A.32: Informazioni sui contatti

Favorites		
Risorsa	**Descrizione**	**Parametri**
flickr.favorites.getList	Restituisce la lista delle foto preferite dall'utente.	ID_photo, ID_owner

Tabella A.34: Dati delle foto preferite

Galleries		
Risorsa	**Descrizione**	**Parametri**
flickr.galleries.getList	Restituisce la lista delle gallerie create dall'utente, ordinate da quella più nuova a quella più vecchia.	ID_gallery, ID_owner, url, date_create, count_photo, title, description.

Tabella A.36: Parametri per le gallerie dell'utente

People		
Risorsa	**Descrizione**	**Parametri**
flickr.people. getPublicGroups	Restituisce la lista dei gruppi pubblici di cui l'utente è membro.	ID_group, name.
flickr.people.getInfo	Restituisce informazioni riguardo l'utente collegato.	ID, username, realname, location, photosurl, profileurl.
flickr.people. getPhotos	Restituisce informazioni riguardo le foto caricate dall'utente.	ID_photo, ID_owner, title.

Tabella A.38: Informazioni sull'utente

Photos		
Risorsa	**Descrizione**	**Parametri**
flickr.photos.getInfo	Restituisce i dati relativi a una foto.	ID_photo, ID_owner, username_owner, realname_owner, location_owner, title, description, count_comments, list of tags, author_tags, url_photo.
flickr.photos. comments.getInfo	Restituisce una lista con i commenti di una foto.	ID_comment, ID_author, authorname, link, comment.

Tabella A.40: Parametri delle foto e dei commenti

A.5 LinkedIn

Come era lecito aspettarsi, anche LinkedIn mette a disposizione degli sviluppatori una serie di API[17] che forniscono una semplice e coerente rappresentazione di persone, società, lavori e le interazioni tra di loro. Anche questa volta i dati vengono restituiti in formato JSON.
Di seguito vediamo un esempio di una richiesta fatta al server di LinkedIn dove non facciamo altro che chiedere alcune informazioni sull'utente che ha effettuato l'autenticazione al social network:

```
http://api.linkedin.com/v1/people/~:(first-name,last-name)
```

Vediamo ora una lista dei metodi, e connessioni, che abbiamo utilizzato e i rispettivi dati che abbiamo memorizzato nel database.

People		
Risorsa	**Descrizione**	**Parametri**
api.linkedin.com/v1/people/~	Restituisce i dati relativi all'utente autenticato.	ID, first-name, last-name, headline, location, industry, summary, specialties, associations, honors, interests, languages, phone-number, im-account, date-of-birth, member-url-resources, picture-url, public-profile-url
Connessione	**Descrizione**	**Risultato**
positions	Una lista degli impieghi svolti dall'utente.	ID, title, summary, start-date, end-date, is-current, company.
patents	Lista dei brevetti acquisiti dall'utente.	ID, title, summary, number, date, url.
pubblications	Lista delle pubblicazioni dell'utente.	ID, title, publisher, date, url, summary.
skills	Insieme delle competenze dell'utente	ID, name, proficiency, years.
certifications	Insieme delle certificazioni conseguite dall'utente.	ID, name, authority, start-date, end-date.
educations	Istruzione dell'utente.	ID, school-name, field-of-study, start-date, end-date, degree, activities, notes.

Tabella A.42: Informazioni sull'utente loggato

Groups		
Risorsa	**Descrizione**	**Parametri**
api.linkedin.com/v1/people/~/group-memberships	Restituisce informazioni riguardo ai gruppi di cui l'utente risulta essere membro.	ID, name, short-description, description, category, website-url, site-group-url, num-members.

Tabella A.44: Dati sui gruppi per i quali l'utente è membro

A.6 Digg

Per Digg vale quanto detto fino ad ora. Per ogni chiamata API[18] vengono restituiti in risposta in dati in formato JSON.
Vediamo quali sono le API utilizzate per prelevare i dati degli utenti.

User		
Risorsa	**Descrizione**	**Parametri**
services.digg.com/ 2.0/user.getInfo	Restituisce il profilo di una persona	ID, username, about, name, gender, diggs, location, following, submissions, icon

Tabella A.46: Informazioni del profilo dell'utente Digg

Activity		
Risorsa	**Descrizione**	**Parametri**
services.digg.com/ 2.0/user.getActivity	Restituisce le attività di un utente come i digg o le submissions	ID, type, permalink, description, title, url, diggs, duggs, comments, topic

Tabella A.48: Informazioni sulle attività dell'utente Digg

Comments		
Risorsa	**Descrizione**	**Parametri**
services.digg.com/ 2.0/com- ment.getInfo	Restituisce le informazioni su un commento	ID, text, diggs, down, up, story_id

Tabella A.50: Informazioni sulle attività dell'utente Digg

A.7 Google+

Anche Google mette a disposizione degli sviluppatori una serie di API[19] per accedere ai dati dell'utente. Al momento le API disponibili sono solo quelle per ottenere le informazioni sul profilo dell'utente e sulle sue attività, inteso come post in bacheca.

User		
Risorsa	**Descrizione**	**Parametri**
https://www.googleapis.com/plus/v1/people/userId	Restituisce il profilo di una persona	ID, displayName, description, gender, aboutMe, url, imageUrl, placeLived

Tabella A.52: Informazioni del profilo dell'utente Google+

Activity		
Risorsa	**Descrizione**	**Parametri**
https://www.googleapis.com/plus/v1/people/userId/activities/public	Elenco delle attività per un dato utente	ID, type, title, url, from_name, from_id, attachment_type, attachment_url, attachment_content, attachment_displayName

Tabella A.54: Informazioni sui post dell'utente

Bibliografia

[1] Stefano Benvenuti - Data aggregation and content refining systems.

[2] John O'Donovan and Barry Smyth. Trust in recommender systems. In Proceedings of the 10th international conference on Intelligent user interfaces, 2005.

[3] Toine Bogers and Antal van den Bosch. Collaborative and content-based filtering for item recommendation on social bookmarking websites. In Proceedings of the ACM RecSys'09 Workshop on Recommender Systems & the Social Web, 2009.

[4] Robin van Meteren and Maarten van Someren. *Using Content-Based Filtering for Recommendation.*

[5] Advaith Siddhartha. *Adaptive Interactive Systems.* University of Aberdeen.

[6] http://www.cineca.it/sites/default/files/DataM_it.pdf.

[7] http://www.report.rai.it/dl/docs/1302458627250prodotto_pdf.pdf.

[8] Ralph Gross and Alessandro Acquisti. *Information Revelation and Privacy in Online Social Networks (The Facebook case).* Pre-proceedings version. ACM Workshop on Privacy in the Electronic Society (WPES), 2005.

[9] [RFC5849] E. Hammer-Lahav, Ed., " The OAuth 1.0 Protocol", RFC 3986, April 2010.

[10] RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[11] [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[12] The Cassandra Distribuited Database - Eric Evans - February 7, 2010.

[13] Facebook API - https://developers.facebook.com/docs/reference/api

[14] Twitter API - https://dev.twitter.com/docs/api

[15] YouTube API - https://developers.google.com/youtube

[16] Flickr API - http://www.flickr.com/services/api

[17] LinkedIn API - http://developer.linkedin.com/apis

[18] Digg API - http://developers.digg.com/documentation

[19] Google+ API - https://developers.google.com/+/api