Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Facebook Searcher: un tool per la ricerca di contenuti all'interno del social network

P. Sangiorgi, D. Terrana, A. Augello, G. Pilato

Rapporto Tecnico N.:
RT-ICAR-PA-13-01 **Dicembre 2013**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Facebook Searcher: un tool per la ricerca di contenuti all'interno del social network

P. Sangiorgi [1], D. Terrana[1], A. Augello [1], G. Pilato [1]

Rapporto Tecnico N.:
RT-ICAR-PA-13-01

Data:
Dicembre 2013

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.

Facebook Searcher: un tool per la ricerca di contenuti all'interno del social network

Pierluca Sangiorgi, Diego Terrana, Agnese Augello, Giovanni Pilato
ICAR - Istituto di Calcolo e Reti ad Alte Prestazioni
CNR - Consiglio Nazionale delle Ricerche
Viale delle Scienze - Edicio 11 - 90128 Palermo, Italy
Email: {sangiorgi, terrana, augello, pilato}@pa.icar.cnr.it

***Abstract*—Nel presente documento viene descritta l'architettura software di un sistema per l'indicizzazione e la ricerca di contenuti all'interno del social network Facebook.**

Keywords: social network; indicizzazione; solr; facebook, rest, soap, graph API

I. INTRODUZIONE

Negli ultimi anni si sta assistendo ad un'evoluzione nel mondo del World Wide Web dovuta alla nascita e ad una crescita esponenziale nell'uso dei social network.
Un social network è una struttura sociale costituita da un insieme di attori (individui o organizzazioni) connessi tra loro da diversi legami sociali (amicizie, rapporti di lavoro, vincoli familiari, etc...), organizzata attorno a una o più aree tematiche oppure sviluppata su base territoriale.
I social network coinvolgono tutte le varie tipologie di utenza associabili al World Wide Web, sia private che pubbliche. Realtà come "Facebook" [1] e "Twitter" [2] si sono imposte convincendo utenti di tutto il globo a iniziare una "grande rivoluzione dei rapporti sociali". Le discussioni tra gli utenti dei social network sono potenzialmente di grande interesse. Milioni di utenti, infatti, in tutto il mondo condividono tramite i social network informazioni, notizie, eventi generali.
Facebook rappresenta al giorno d'oggi il social network di maggior successo a livello mondiale e, nella sua continua evoluzione di gestione dei contenuti, sta perseguendo l'obiettivo di accentrare in un unico sistema informativo, informazioni provenienti da più sistemi informativi esterni ed eterogenei fra loro, in modo da fornire all'utente un'unica piattaforma centralizzata di consultazione delle informazioni strutturate in maniera omogenea.
Tale strategia implica che l'utilizzatore non è più costretto a navigare sulla rete alla ricerca di informazioni di suo interesse, poiché si trova ad utilizzare una piattaforma che gli suggerisce gli argomenti potenzialmente di suo interesse sulla base della profilazione che il sistema è in grado di effettuare e della condivisione dei contenuti all'interno della sua rete sociale.
Il risultato finale è una crescita esponenziale della quantità dei dati generata e la frequenza con cui vengono prodotti all'interno di Facebook e si fa sempre più forte l'esigenza di sistemi efficienti nella ricerca dei contenuti in quanto ad oggi per gli utenti è pressoché impossibile effettuare una ricerca sui propri contenuti se non scorrendoli manualmente.
Facebook mette a disposizione uno strumento per la ricerca dei contenuti al proprio interno, tuttavia esso è caratterizzato da un linguaggio ostico sia nella fase di interrogazione che in quella di produzione dei risultati, producendo in uscita una mole considerevole di dati ma di difficile comprensione. Facebook infatti non fornisce strumenti di interrogazione e produzione dei risultati tramite interfacce grafiche di semplice utilizzo, in quanto l'uso dello strumento di ricerca è pensato per l'utenza degli sviluppatori e non per quella end user, che di fatto restano senza un adeguato strumento di ricerca.
Si è quindi cercato di sopperire a tale mancanza, creando un sistema in grado di fornire agli utenti Facebook la funzionalità di ricerca dei contenuti grazie alle esperienze acquisite nel corso dei progetti PON01_01687 - SINTESYS (Security and INTElligence SYstem) e POR 2007/2013 - Regione Siciliana - Misura 4.1.1.1. - IDS (Innovative Document Sharing) rispettivamente nei campi dell'estrazione dei dati dal social network e nei sistemi di indicizzazione e ricerca.
In particolare è stato realizzato un prototipo di sistema che implementa le funzionalità di data mining sulle risorse informative relative agli account Facebook, filtrandone le informazioni di particolare interesse e strutturandole in modo da effettuare operazioni di indicizzazione e successiva ricerca dei contenuti.

II. TECNOLOGIE

Di seguito si riporta una descrizione delle due principali tecnologie fornite da terze parti utilizzate per lo sviluppo del sistema: Facebook Graph Api [3] ed Apache Solr [4].

A. Facebook Graph API

Le Graph API sono il modo principale per inserire ed esportare dati dal grafo sociale di Facebook. Si tratta di API di tipo http di basso livello che è possibile utilizzare per eseguire query sui dati, inserire nuove storie, creare il check-in o uno qualsiasi degli altri compiti che un app potrebbe dover fare. Open Graph di Facebook consente di definire nuovi oggetti e azioni nel grafo sociale di un utente, e il modo in cui si creano nuove istanze di quelle azioni e oggetti avviene tramite l'API Graph. Il grafo sociale non è una semplice tabella di dati, ma una serie di nodi tutti collegati tra loro. Le API Graph si fondano su richieste HTTP. I metodi HTTP sono associati direttamente con azioni sul grafico. Il metodo GET viene usato per leggere, POST per modificare ed aggiungere mentre DELETE per rimuovere i nodi. Con l'Explorer [5] è possibile
selezionare il metodo (GET, POST o DELETE), il percorso (/ID) e gli eventuali modificatori alla richiesta (fields=id, nome) (Figura 1).

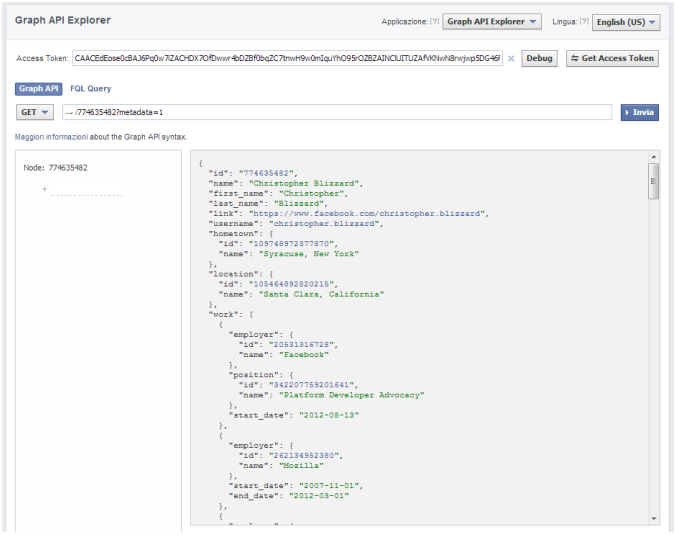

Figura 1: Graph Api Explorer

Ci sono due tipi di oggetti che è possibile accedere su Facebook per nome: gli account Personali di un utente e le Pagine pubbliche. Ogni richiesta che si fa richiede un token di accesso. Il token di accesso è una stringa che viene generata alla fine del processo di autorizzazione. Rappresenta un insieme di autorizzazioni che sono state concesse, e può essere utilizzata nel contesto di una applicazione specifica o una persona specifica. Un token di accesso ha una scadenza. Esistono token di accesso che consentono di lavorare sia con i dati di una persona che con dati di una pagina o un'app. Il 'page token' viene utilizzato per gestire i dati di per una specifica pagina di Facebook mentre un 'app token' consente di accedere ai dati specifici di un'app [6]. Un nodo speciale nel grafo è quello che si riferisce all'utente corrente. Il percorso da usare in questi casi è '/me' e permette di ottenere tutte le informazioni relative all'utente loggato (Figura 2).
E' possibile vedere alcuni dati come: nome, luogo, storia, alcune informazioni sui favoriti, sull'istruzione, ecc. Molti degli oggetti nei dati restituiti sono link che è possibile esplorare.
Aggiungendo alla richiesta HTTP il parametro per l'abilitazione dei metadati ('metadata =1'), saranno visibili ulteriori informazioni relative ai metadati per i campi che sono disponibili, nonché una serie di connessioni. Tutte queste connessioni sono collegamenti ipertestuali, ed è sufficiente fare clic su di loro per caricarli.

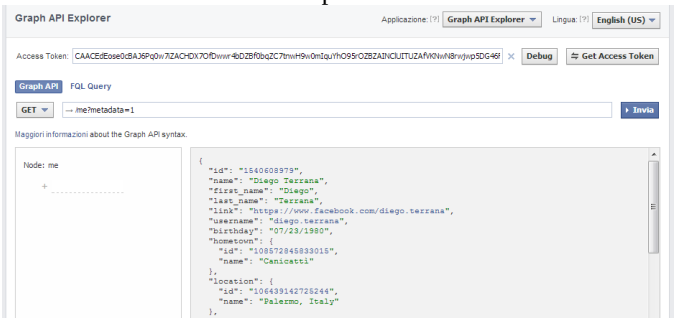

Figura 2: Graph Api Explorer, percorso speciale '/me'

Cliccando sul link individuato da 'statuses' sarà possibile visualizzare tutti ciò che l'utente ha postato. Le principali informazioni visualizzabili, sempre secondo le norme stabilite dagli autori di Facebook sulla privacy ed accettate dall'utente, sono: Home page, Friends, Family, Activities, Interests, Music, Books, Movies, Post, Links, Notes, Groups, Posts, Photos, Likes e Statuses.
Un esempio di semplice e flessibile Facebook Graph API disponibile in rete è RestFB [7]. Si tratta di software open source rilasciato sotto i termini della licenza MIT. L'obiettivo è quello di interagire con Facebook per acquisire i dati dell'utente corrente o di una pagina pubblica. Per invocare I servizi Facebook occore un access token valido.
Prima di tutto occorre creare un client per l'accesso a Facebook. A tal scopo s'istanzia la classe **FacebookClient** il cui costruttore riceve in ingresso la stringa contenente l'access token:

```
FacebookClient facebookClient =
new DefaultFacebookClient (accessToken);
```

Per ottenere le informazioni relative all'utente corrente, si invoca il metodo **fetchObject**, passando come parametri, rispettivamente, il nome dell'utente e la classe corrispondente alle informazioni richieste:

```
User user =
facebookClient.fetchObject("mario.rossi"
, User.class);
```

DefaultFacebookClient è l'implementazione del Facebook Client fornito con RestFB.
Per motivi di performance, è possibile anche limitare la chiamata ad un gruppo ridotto di informazioni:

```
User user =
facebookClient.fetchObject("mario.rossi,
User.class, Parameter.with("fields",
"id, name"));
```

In questo caso, gli unici parametri della classe **User** che troveremo valorizzati saranno id e name. Tutti gli altri risulteranno vuoti.
Se invece si vuole richiedere le informazioni di una Pagina, si mappa il tutto con la classe **Page**:

```
Page pageWithMetadata =
facebookClient.fetchObject(page,
Page.class,Parameter.with("metadata",1))
```

Per ottenere tutti i post di una pagina occorre utilizzare il percorso '/feed' :

```
Connection<Post> myFeed =
facebookClient.fetchConnection(page.getI
d()+"/feed", Post.class);
```

B. Apache Solr

Apache Solr è un server di ricerca open-source basato sulle librerie Apache Lucene e distribuito sotto licenza "*Apache License, Version 2.0*".

Internamente è sviluppato in linguaggio di programmazione Java, ma consente un utilizzo indipendente dalla piattaforma poiché permette di accedere alle operazioni di inserimento (indexing) e ricerca (searching) sui documenti tramite api REST-like http/XML [8] e JSON [9].
Unico requisito per il suo utilizzo è l'impiego di un servlet container come Apache Tomcat per la sua esecuzione.

1) Principi basilari

Apache Solr gestisce i dati al proprio interno impiegando un repository basato su una struttura ad indice invertito, ovvero composta da due elementi: il vocabolario e le occorrenze. Il vocabolario è costituito dall'insieme di tutte le parole del testo a cui vengono associate la lista delle posizioni nel testo dove esse compaiono. Tali liste costituisco le occorrenze.
In Solr e Lucene, un indice è costituito da uno o più *Document* ed ognuno consiste di uno o più *Field*.
Un *Field* è costituito da un nome, il contenuto ed una serie di parametri che specificano al sistema come trattarne il contenuto. Ad esempio i campi possono contenere stringhe, numeri, valori booleani, date, ecc, ed ogni campo può essere descritto usando determinate opzioni che specificano al sistema come trattarne il contenuto durante le fasi di indicizzazione e ricerca (processo di analisi).

2) Processo di analisi

Se opportunamente specificato, Solr avvia il processo di analisi sui campi di un documento durante le fasi di indicizzazione e/o ricerca. Esempio di processi di analisi sono:

- Eliminazione delle stopword: articoli, congiunzioni ecc.;
- De-hyphenation: divisione in più token di parole contenenti un trattino;
- Stemming: riduzione delle parole alla loro radice grammaticale;
- Thesauri: gestione dei sinonimi

3) Operazioni di indicizzazione

L'indicizzazione è il processo durante il quale Solr riceve in input *Document* inviati attraverso messaggi HTTP POST.
E' possibile inviare quattro differenti tipi di richieste di indicizzazione:

- **add/update** consente di aggiungere o modificare documenti in Solr. Le modifiche al sistema non saranno disponibili per la ricerca finchè non viene effettuato il commit.
- **commit** specifica che le ultime modifiche apportate al sistema devono essere rese disponibili per la ricerca.
- **optimize** ristruttura i file di Lucene al fine di migliorare le performance durante la ricerca
- **delete** elimina un document specificato da id o un insieme di documenti risultati da una query.

4) Comandi di ricerca

Solr accetta messaggi HTTP GET ed HTTP POST per le query di ricerca che vengono processate da apposite istanze dell'interfaccia SolrRequestHandler.
La sintassi per le query in Solr è la stessa supportata da LuceneQueryParser, con l'aggiunta di funzionalità per l'ordinamento.
Un esempio di semplice query Solr è la seguente:

http://localhost:8983/solr/select?q=myField:Java AND otherField:developerWorks; date asc

con la quale si ricercano i termini "Java" e "developerWorks" rispettivamente nei campi "myField" e "otherField" e si ordina il risultato sulla base del campo "date".
Una volta sottomessa la query, in caso di match con i documenti indicizzati, Solr restituisce una risposta XML contentente i risultati.
Solr consente inoltre di effettuare ricerche sfaccettate su specifici Fields opportunamente configurati.

5) Il Solr schema

Solr gestisce le funzionalità descritte sulla base dell'apposito file di configurazione *schema.xml*.
Lo schema è fondamentale per la definizione della struttura che devono avere i documenti trattati all'interno del server ed è suddiviso in due sezioni principali:

- Types
- Fields

Nella sezione *fields* vengono definiti i campi che compongono i documenti, specificando per ognuno di essi un nome, la tipologia (*types*) e se devono essere indicizzati e memorizzati all'interno del sistema.
Nella sezione *types* vengono definite le tipologie che possono assumere i *fields*. Tali tipologie possono essere semplici stringhe, valori interi, ecc, oppure tipi di dati più complessi in cui viene espressamente indicato il processo di analisi che deve essere applicato al contenuto dei campi appartenenti a quella tipologia per l'estrazione dei singoli termini.
Il processo di analisi può essere definito indipendentemente per le due diverse fasi di indicizzazione e ricerca e prevede la definizione di un *tokenizer* per l'estrazione dal contenuto di token rispondenti a determinate regole, più tutta una serie di filtri quali eliminazione di stopword, stemming, etc.
Il sistema quindi indicizzerà per ogni campo tutti i termini risultanti dal processo di analisi definito nello specifico tipo definito per quel campo ed analogamente effettuerà le interrogazioni con i termini risultanti dall'altro processo di analisi definito per la fase di ricerca.

III. ARCHITETTURA DEL SISTEMA

La figura 3 mostra l'architettura del sistema proposto. Si tratta di un prototipo di sistema che esegue il data mining di risorse informative relative agli account Facebook, filtrandone le informazioni di particolare interesse e strutturandole in modo da effettuare operazioni di ricerca.

Il sistema espone tali funzionalità tramite servizi web che possono quindi essere fruite dall'utilizzatore finale tramite un semplice client realizzato ad hoc al fine di rendere efficiente l'esperienza d'uso nello specifico campo di utilizzo. L'utente accede tramite un punto di accesso alle funzionalità del sistema. Egli, utilizzando le funzionalità fornite dal modulo 'Facebook Manger', può avviare una scansione di una pagina recuperando dal grafo tutte le informazioni disponibili secondo le policy di sicurezza e privacy definite da Facebook. Tali informazioni saranno opportunamente indicizzate e memorizate dal modulo *'Solr Manager'*. L'utente può anche effettuare ricerche sulle infromazioni precedentemente memorizzate. Le seguenti sezioni descrivono nel dettaglio le diverse parti.

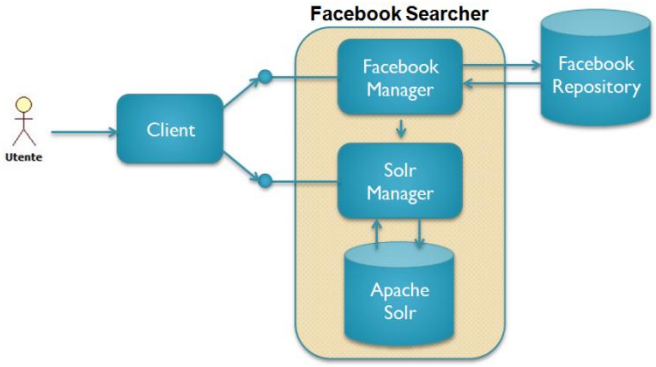

Figura 3: Architettura del sistema Facebook Searcher

Tutti i software e le librerie usate nello studio e nella realizzazione del prototipo sono open source. Il linguaggio scelto per implementare il prototipo è Java in quanto presenta caratteristiche particolari che bene si adattano alle problematiche del progetto.
In particolare tale linguaggio è indipendente dall'architettura hardware e software sottostante, il codice è portabile grazie alla presenza della Java Virtual Machine su tutti i principali sistemi operativi presenti sul mercato (Windows, Linux, MacOS).
Inoltre è un linguaggio orientato agli oggetti, quindi oltre a fornire un'efficiente implementazione ed una buona pulizia stilistica, permette un'eventuale estensione del sistema con nuove funzionalità in maniera efficiente grazie alle tecniche di modularizzazione.

A. Facebook Manager

Facebook Manager è un modulo che espone dei servizi che consentono di recuperare tutti i post di Facebook relativi ad una pagina o ad un utente, di filtrare e formattare le informazioni raccolte secondo le specifiche definite nel modulo di indicizzazione(*Solr Manger*). Tali dati vengono estratti utilizzando le API fornite da RestFB. Le informazioni estraibili sono solo tutte quelle permesse dalle policy di privacy di Facebook. La richiesta PUT deve contenere lo *username* o l'*id* dell'utente/pagina che si vuole analizzare in un campo di nome *'facebookId'* e l'accessToken da usare per gli accessi a Facebook in un campo di nome *'accessToken'*. Il servizio si occupa anche della gestione della paginazione dei risultati. Facebook, infatti, limita il numero di post visualizzabili contemporaneamente su una pagina.
Al fine di recuperare tutti i post pubblicati su una pagina si utilizzano le informazioni contenute nei parametri "paging_token", "until" e "limit" [10].
Facebook Manager definisce un task che controlla periodicamente i dati aggiornando i post che contengono nuove informazioni. Tutte le informazioni estratte vengono incapsulate in oggetti di tipo *Document* ed inviate al modulo *'Solr Manager'* per l'indicizzazione. Le informazioni indicizzate per ogni post sono le seguenti:

- id post(**ID**);
- id utente autore del post(**FROM**) ;
- data creazione post(**CREATEDDATE**);
- elenco commenti(**COMMENT**);
- tipo post: testo, video, audio, immagine...(**TYPE**);
- messaggio post(**MESSAGE**);
- data ultima modifica(**UPDATEDATE**);
- link alla risorsa sul grafo Facebook(**LINK**).

Facebook inpone alcune limitazioni sul numero di commenti o like recuperabili tramite API. In entrambi i casi non è possibile recuperare le informazioni di più di 25 *'like'* o *'comment'*. Per superare parzialmente queste limitazioni, il servizio accede accede direttamente alle tabelle 'LIKE' , 'USER' e 'COMMENT' di Facebook utilizzando *Facebook Query Language (FQL) Table Reference* [11].
Le query generate per i commenti ed i like sono rispettivamente del tipo:

```
SELECT [commentField] FROM Comment WHERE post_id='idPost'
```

```
SELECT [likeField] FROM Like WHERE post_id ='idPost'
```

Utilizzando tali query è possibile superare il limite di 25 sino ad arrivare a 100 risultati.
Per estrarre le relative informazioni degli utenti che commentano o mettono un like ad un post la query usata è la seguente:

```
SELECT [userField] FROM User WHERE uid ='userId'
```

Il risultato fornito presenta le informazioni estratte in formato JSON (Figura 4).

```
{"Posts":[
{"Caption": "", "Place": {},"Description": "",
   "CreatedTime": "Fri Oct 18 02:47:19 CEST 2013",
   "Message": "", "Type": "link",
   "messageTags":[], "likesCount": "", "To": {},
   "UpdatedTime": "Fri Oct 18 02:49:20 CEST 2013",
   "id": "100000503162381_741986625828103",
   "Name": "","Source": "",
   "usersLike": [],
   "comments": [{
              "id": "741986625828103_32094960",
              "LikeCount": 1, "Type": "",
              "CreatedTime": "Fri Oct 18 02:49:20 CEST 2013",
              "Message": "Si parte dall' APERITIVO....",
              "likes": [ {
                        "sex: ": "male",
                        "first_name: ": "Danilo",
                        "last_name: ": "Pensabene"
                        "name": "DaniloPensabene",
                        "birthday_date: ": null,
                        "middle_name: ": "danilo.pensabene",
                        "username: ": "danilo.pensabene",
                         },
                       ],
             "UserLikes": false,
             "from": {
                    "Name": "DaniloAndrea",
                    "Category": "", "Type": "",
                    "Id": "100001321111710"
                    }
          }],
   "From": {
          "Name": "PalermoDiNotte",
          "Category": "",
          "Type": "",
          "Id": "100000503162381"
         }
```


Figura 4: Schema di esempio dei risultati in notazione JavaScript Object Notation(JSON).

Il sistema è stato modulato seguendo lo stile architetturale REST per il disegno di applicazioni di rete secondo il protocollo di comunicazione stateless che consente una comunicazione tra macchine basata su richieste standard http, indipendenti dalla piattaforma, indipendenti dal linguaggio di programmazione senza particolari configurazioni di firewall.

B. Solr Manager

Solr Manager è un modulo che espone dei servizi che consentono di accedere alle funzionalità di indicizzazione e ricerca del server Apache Solr. Svolge quindi lo scopo di interconnettere il modulo Facebook Manager dal quale riceve i dati relativi all'utente, ed un'istanza di Apache Solr configurata con gli opportuni campi per gestire quei dati che verranno memorizzati all'interno dell'apposito repository costituito da indici in formato Apache Lucene.

L'interconnessione viene implementata esponendo dei metodi per l'aggiunta, la modifica e la rimozione dei *Document* di Apache Solr.

Tali metodi ricevono in ingresso dal Facebook Manager un'HashMap contenente le coppie (campo, valore) che definiscono i dati d'interesse estratti dal file json. Tali dati vengono poi strutturati in un oggetto di tipo *Document* conforme alla struttura definita nello schema (Figure 5 e 6) dell'istanza di Apache Solr, e lo inviano a quest'ultima, tramite la libreria Apache SolrJ Client.

Fatto ciò i Document verranno indicizzati all'interno del repository a seguito del loro trattamento a livello di processo di analisi così come definito nel file schema.

Il risultato di tali operazioni sarà un valore booleano che indica il successo o meno della funzionalità richiesta dal Facebook Manager che potrà quindi procedere con altre richieste oppure re inoltrare la stessa in caso di esito negativo.

La funzionalità di ricerca dei contenuti viene svolta invece interconnettendo un apposito client sviluppato per l'interrogazione ed il server Apache Solr.

In questo caso il sistema riceve in ingresso un'HashMap contenente le coppie (campo, valore) che indicano su quali campi e con che valori effettuare l'interrogazione ed un operatore logico che ha il compito di definire come queste chiavi vanno combinate fra loro per la composizione dell'interrogazione.

Una volta ricevuti i parametri di ricerca, sempre tramite utilizzo della libreria Apache SolrJ Client, viene composta con un "*query builder*" l'interrogazione aderente allo standard Solr e sottoposta sul repository dei documenti indicizzati fino a quel momento.

Il risultato di tale interrogazione sarà a sua volta formattato all'interno di un'HashMap contenente le coppie (id del documento, valore di pertinenza con l'interrogazione) ed inoltrato al client che ne ha fatto richiesta e che li riprodurrà nella maniera più idonea.

```
<fieldType name="text_ita" class="solr.TextField"
positionIncrementGap="100">
            <analyzer type=”index”>
                        <tokenizer class="solr.StandardTokenizerFactory"/>
                        <filter class="solr.LowerCaseFilterFactory" />
                        <filter class="solr.ElisionFilterFactory"
                        ignoreCase="true”
                                articles="lang/contractions_it.txt" />
                        <filter class="solr.StopFilterFactory"
                        ignoreCase="true"
                                 words="lang/stopwords_it.txt"
                        format="snowball"
                                enablePositionIncrements="true" />
                        <filter class="solr.SnowballPorterFilterFactory"
                                language="Italian"/>
            </analyzer>
            <analyzer type="query">
                        <tokenizer class="solr.StandardTokenizerFactory"/>
                        <filter class="solr.LowerCaseFilterFactory" />
                        <filter class="solr.ElisionFilterFactory"
                        ignoreCase="true"
                                articles="lang/contractions_it.txt" />
                        <filter class="solr.StopFilterFactory"
                        ignoreCase="true"
                                words="lang/stopwords_it.txt"
                        format="snowball"
                                enablePositionIncrements="true" />
                        <filter class="solr.SnowballPorterFilterFactory"
                        language="Italian"/>
            </analyzer>
</fieldType>
```

Figura 5: Sezione types del Solr Schema

C. Client

Il Client rappresenta il punto di accesso per l'utilizzatore finale alle funzionalità offerte dal sistema.

Tramite questo modulo l'utente accede al sistema inserendo le proprie credenziali, dalle quali il modulo Facebook Manager recupera il relativo access token ed avvia il processo di estrazione dei dati relativi all'utente.

Una volta creato il proprio archivio informativo, l'utente può in qualunque momento effettuare interrogazioni tramite parole chiave sui diversi campi che compongono i documenti indicizzati, effettuando anche composizioni fra chiavi al fine di creare query più complesse, restringere il numero dei documenti ed ottenere risultati plausibilmente più pertinenti.

Il client riceve dal sistema il risultato prodotto da tali interrogazioni e si occupa di presentarlo all'utente in maniera adeguata alla fruizione dei risultati.

```
<fields>
        <field name="id" type="string" indexed="true"
        stored="true" required="true" />
        <field name="type" type="string"
        indexed="true" stored="true" />
        <field name="createdDate" type="string"
        indexed="true" stored="true" />
        <field name="updatedDate" type="string"
        indexed="true" stored="true" />
        <field name="link" type="string"
        indexed="true" stored="true" />
        <field name="from" type="text_ita"
        indexed="true" stored="true" />
        <field name="message" type="text_ita"
        indexed="true" stored="true"
        termVectors="true" />
        <field name="comment" type="text_ita"
        indexed="true" stored="true"
        termVectors="true" />
</fields>
```

Figura 6: Sezione fields del Solr Schema

Il client, grazie all'architettura del sistema orientata ai servizi, è indipendente dalla piattaforma e non è vincolato da una particolare architettura. Unico requisito è quello di integrare un service endpoint per l'accesso ai servizi esposti dal sistema che definisce la sua interfaccia di comunicazione.
Il client è quindi una generica classe di applicativo che può essere istanziata molteplici volte in differenti tipologie quali ad esempio un'applicazione web, stand-alone, per dispositivi mobili, etc, in base allo specifico utilizzo richiesto per il particolare utente, differenziando le modalità di interazione uomo macchina, pur mantenendo le stesse funzionalità.

IV. CONCLUSIONI

Dopo la realizzazione del prototipo sono state effettuate diverse sessioni di prova da parte di più utenti su diversi contenuti.
Il risultato dall'esperienza d'uso è stato un sistema dimostratosi efficiente nel produrre risultati pertinenti con le interrogazioni degli utenti ed ha evidenziato la sua versatilità nel poter essere utilizzato per diversi scopi, quali ad esempio recupero d'informazione da pagine di personaggi politici ed organizzazioni per la creazione di dataset relativi ad un particolare dominio, piuttosto che ricerca di contenuti da utenti end-user all'interno della propria pagina.
Prestazionalmente l'utilizzo di Apache Solr con la sua struttura ad indice invertito si riflette in tempi di risposta da parte del sistema pressoché immediati (nell'ordine dei millisecondi) durante la fase di interrogazione per la ricerca dei contenuti. I maggiori tempi di attesa del sistema risiedono nella precedente fase di estrazione ed indicizzazione dei dati, dove il tempo impiegato nell'operazione è fortemente dipendente e direttamente proporzionale alla quantità di dati che si vogliono elaborare. Il tempo di attesa maggiore in tali fasi si verifica comunque solo in occasione della scelta dell'account d'interesse e prima indicizzazione dei contenuti, mentre successivamente i tempi sono relativi ai nuovi dati inseriti nel periodo tra un'estrazione e la successiva che sono in ogni caso operazioni effettuabili in background tramite schedulazione senza la necessità dell'intervento dell'utente.
I tempi di risposta rapidi rendono il sistema altamente usabile. All'utente non è richiesta una formazione specifica, egli è in grado di navigare recuperando facilmente i contenuti. L'informazione è presentata in modo chiaro e conciso, riducendo gli sforzi cognitivi dell'utente. L'obiettivo raggiunto è quello di rendere invisibile all'utente la tecnologia sottostante che può così concentrarsi solo sull'oggetto della ricerca.

REFERENCES

[1] https://www.facebook.com/
[2] https://twitter.com/
[3] https://developers.facebook.com/docs/getting-started/graphapi/
[4] http://lucene.apache.org/solr/
[5] https://developers.facebook.com/tools/explorer
[6] https://developers.facebook.com/docs/facebook-login/access-tokens/
[7] http://restfb.com/
[8] http://en.wikipedia.org/wiki/Representational_state_transfer
[9] http://www.json.org/
[10] https://developers.facebook.com/docs/reference/api/search/
[11] https://developers.facebook.com/docs/reference/fql/