Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

ASSOCIAZIONI NON PROFIT E LINKED OPEN DATA: UN ESPERIMENTO

Relatore:
Chiar.mo Prof.
ANGELO DI IORIO

Co-relatore:
FRANCESCO POGGI

Presentata da:
GIULIA FRANCHINI

Sessione III
Anno Accademico 2013/2014

Knowledge speaks, but wisdom listens...

JIMI HENDRIX

Introduzione

Le Associazioni Non Profit giocano un ruolo sempre più rilevante nella vita dei cittadini e rappresentano un'importante realtà produttiva del nostro paese; molto spesso però risulta difficile trovare informazioni relative ad eventi, attività o sull'esistenza stessa di queste associazioni. Secondo una ricerca [16] condotta da Google nel 2013, Internet rappresenta il principale punto di riferimento per le persone che cercano informazioni riguardo queste realtà, ma il Web è diventato ad oggi un'immenso oceano di dati che risulta difficile da navigare senza avere a disposizione i giusti strumenti con i quali orientarsi.

Per questo motivo molte Regioni e Province mettono a disposizione degli elenchi in cui sono raccolte le informazioni relative alle varie organizzazioni che operano sul territorio. Questi elenchi, che dovrebbero venire in contro alle esigenze dei cittadini, presentano spesso grossi problemi, quali: dati mancanti, informazioni non corrette, pochi dati a disposizione e ambiguità. Se analizziamo la questione da un punto di vista tecnico, saltano fuori ulteriori problemi legati ai formati e alle licenze tramite le quali sono solitamente rilasciati; ad esempio, il fatto che i dati siano pubblicati per mezzo di file PDF o pagine HTML, non li rende machine-readable, il che significa che pur essendo liberamente consultabili e comprensibili all'uomo, non possono però essere facilmente interpretati e processati automaticamente da delle macchine: questo definisce dei grossi limiti sul loro utilizzo, in quanto diventa molto difficile pensare di utilizzarli come base su cui sviluppare e fornire dei servizi.

Tutti questi fattori hanno portato all'idea e alla necessità di realizzare

un sistema per raccogliere, sistematizzare e rendere fruibili le informazioni sulle Associazioni Non Profit presenti sul territorio, in modo che questi dati possano essere utilizzati liberamente da chiunque per scopi diversi.

Il lavoro svolto nell'ambito di questa tesi prende il nome di *'Progetto OnDa'* e si pone due precisi obiettivi; il primo consiste nell'implementazione di un tool in grado di recuperare le informazioni sulle Associazioni Non Profit sfruttando i loro Siti Web. Al giorno d'oggi Internet permette alle organizzazioni di interagire con un pubblico molto vasto con costi limitati, non solo utilizzando i classici Siti Web, ma anche attraverso l'utilizzo dei Social Network che hanno raggiunto negli ultimi anni una larghissima diffusione in tutto il paese e in tutte le fasce di età. Per questi motivi, nel 2015, ci si aspetta che la maggior parte delle associazioni possieda un Sito Web o un profilo social che le rappresenti e ne descriva le caratteristiche essenziali, come i motivi dell'esistenza, la collocazione sul territorio e le attività svolte. L'utilizzo di questi canali come mezzo per reperire le informazioni cercate risulta quindi ottimale, sia per quanto riguarda la correttezza dei dati (ci si aspetta infatti che le informazioni contenute sui siti personali siano costantemente aggiornate dalle associazioni), sia per quanto riguarda la possibilità di raccogliere grandi quantità di dati in tempi ragionevoli e nel modo più automatizzato possibile.

Il secondo obiettivo del progetto OnDa consiste nel pubblicare le informazioni raccolte, secondo dei modelli che ne permettano un uso libero e non vincolato. Nello specifico, quello che si vuole ottenere è un dataset che possa essere utilizzato per sviluppare dei servizi e creare delle applicazioni; per questo, innanzitutto, è necessario che i dati siano pubblicati in modo 'aperto', seguendo le specifiche degli Open Data. Il movimento Open Data prevede infatti il rilascio di dataset che possono essere utilizzati liberamente e redistribuiti da chiunque, con l'unico vincolo di condividerli con lo stesso tipo di licenza con la quale sono stati originariamente rilasciati. Però l'apertura da sola non basta. È importante che i dati siano strutturati e modellati secon-

do degli standard che ne permettano un utilizzo semplice, sia per gli esseri umani sia per i software automatici. Quest'aspetto è fortemente collegato all'idea del Semantic Web. Il Web Semantico rappresenta infatti il Web in grado di mostrare i propri contenuti in modo che possano essere comprensibili facilmente anche da una macchina e non solo dall'uomo.

I concetti appena introdotti rappresentano il contesto di riferimento in cui si inserisce il lavoro svolto e saranno approfonditi ampiamente nel corso della trattazione.

Per quanto riguarda la realizzazione del primo obiettivo e l'implementazione dei tool si è pensato di sfruttare le tecniche di Web Crawling e Web Scraping, rispettivamente per ricercare all'interno del WWW gli URL dei Siti Web delle associazioni e successivamente estrapolare le informazioni di interesse.

In riferimento al secondo obiettivo e la pubblicazione del dataset, si è scelto di utilizzare un modello basato sui principi dei Linked Open Data. I dati pubblicati seguendo le direttive dei LOD offrono molti benefici sia per gli utenti che per gli sviluppatori e presentano numerosi vantaggi, quali alta interoperabilità, riusabilità e possibilità di combinare insieme dati provenienti da fonti diverse per arricchire e generare nuova informazione. Nello specifico, per questo lavoro di tesi e la creazione del dataset, è stato preso in considerazione principalmente il mondo del Non Profit a livello della provincia di Bologna, in quanto si tratta di una realtà sulla quale potevo avere un maggior controllo e che mi permetteva di poter effettuare in maniera molto semplice delle verifiche sui risultati ottenuti, in modo da avere un riscontro sulla qualità del lavoro svolto.

La seguente trattazione è strutturata in questo modo: nel primo capitolo viene fornita una panoramica sullo scenario di riferimento; nello specifico, sono introdotti nel dettaglio il problema e l'idea per una possibile soluzione. Succesivamente sono fornite le nozioni di base relative ai concetti fondamentali legati al lavoro svolto e che risultano necessari per avere un quadro

completo del contesto: Web Crawling, Web Scraping e Linked Open Data.

Nel secondo capitolo è presentato il progetto OnDa e sono forniti i dettagli relativi al suo utilizzo; è inoltre presente una descrizione dettagliata della struttura del dataset creato e su come poter consultare i dati raccolti.

I dettagli tecnici e implementativi sono invece presentati nel terzo capitolo, nel quale sono indicate le varie tecnologie utilizzate e sono riportati esempi di codice utili per capire come sono stati progettati e realizzati i tools.

Il quarto capitolo descrive nello specifico il caso di studio considerato e contiene un'analisi del dataset prodotto (bontà dei dati raccolti, confronto con gli elenchi e i materiali già esistenti) e le problematiche principali riscontrate durante la raccolta dei dati.

Infine, nelle conclusioni, viene espresso un giudizio personale in merito al lavoro svolto e sono forniti alcuni dettagli riguardo a possibili sviluppi futuri legati al progetto realizzato.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Associazioni Non Profit e Linked Open Data

1.1 Introduzione al problema

Al 31 Dicembre 2011 (data dell'ultimo censimento effettuato a livello Nazionale) le Organizzazioni e Associazioni Non Profit attive in Italia risultano 301.191[1]. Rispetto alla precedente rilevazione censuaria del settore, i dati raccolti hanno registrato un aumento del 28% e questi numeri sono destinati a crescere sempre di più con il passare degli anni.

Trovandoci di fronte a numeri così grandi, non è difficile immaginare le difficoltà che possono nascere per un utente che vuole effettuare una ricerca riguardo le Associazioni Non Profit, con lo scopo di reperire e sintetizzare le informazioni di interesse. Per questo motivo, i Comuni e le Regioni spesso mettono a disposizione degli elenchi nei quali sono raccolte le informazioni relative alle varie Organizzazioni che operano sul territorio. Questi elenchi sono solitamente consultabili tramite dei Siti Web, o pubblicati per mezzo di file in formato PDF[1].

Se prendiamo come riferimento il mondo del Non Profit a livello della re-

[1]Portable Document Format

gione Emilia-Romagna si contano all'incirca 25.000 associazioni, il 20% delle quali solamente nella provincia di Bologna (sempre secondo i dati raccolti dall'*ISTAT*[2]). Gli elenchi pubblicati e messi a disposizione dal Comune e dalla Regione, sono consultabili online rispettivamente ai seguenti indirizzi:

- http://www2.provincia.bologna.it/associaz.nsf/
- https://wwwservizi.regione.emilia-romagna.it/teseofe/associazioni-promozione-sociale.asp

Analizzando gli elenchi, sono saltate subito all'occhio diverse problematiche. La prima riguarda il numero di Associazioni in elenco (circa 750), che confrontato con il numero totale di Associazioni presenti sul territorio, risulta molto basso. Ho inoltre riscontrato molte imprecisioni e incongruenze, che mettono in dubbio la correttezza dei dati forniti. Ad esempio ho notato la presenza in elenco di associazioni che sembrano non più attive, in quanto effettuando delle ricerche online non si riesce a risalire a nessuna notizia o informazione ad esse associate, oppure gli unici risultati ottenuti risalgono a molti anni fa, il che fa pensare che l'organizzazione non esista più. In altri casi, le informazioni relative ai contatti, o alla località (indirizzo) sono risultate non coincidenti con quelle contenute nei siti web delle associazioni (vedi figura 1.1).

Tutto questo porta a pensare che le informazioni contenute in questi elenchi non siano aggiornate frequentemente e questo rappresenta un grosso problema, in quanto può creare confusione e fornire informazioni errate agli utenti.

Altri fattori che avvalorano queste considerazioni riguardano i link che dovrebbero rimandare alle pagine web delle associazioni. Per molte associazioni non è indicato alcun sito web di riferimento, ma effettuando delle semplici ricerche online si scopre che in realtà i siti esistono, ma non sono riportati (vedi figura 1.2); inoltre spesso i link indicati tra le informazioni rimandano a pagine web che non sono più raggiungibili, perchè il sito non

[2]http://www.istat.it/

Figura 1.1: Esempio di informazioni errate riguardo l'indirizzo

esiste più o magari è stato spostato su un altro dominio e fa quindi riferimento ad un URL diverso (vedi figura 1.3). In un mondo in cui Internet è diventato uno dei maggiori, se non il principale punto di riferimento attraverso il quale divulgare e reperire informazioni, errori e mancanze di questo genere rappresentano dei limiti importanti.

Figura 1.2: Esempio in cui il sito web non è indicato ma esiste

Si provi a pensare, ad esempio, di voler consultare i Siti Web delle associazioni per recuperare le informazioni riguardo agli eventi, i corsi e le attività da esse organizzate; quest'operazione risulta molto difficile se gli URL presenti in elenco rimandano a siti non aggiornati o non più raggiungibili. Un altro aspetto importante da considerare in questo contesto, riguarda i Social

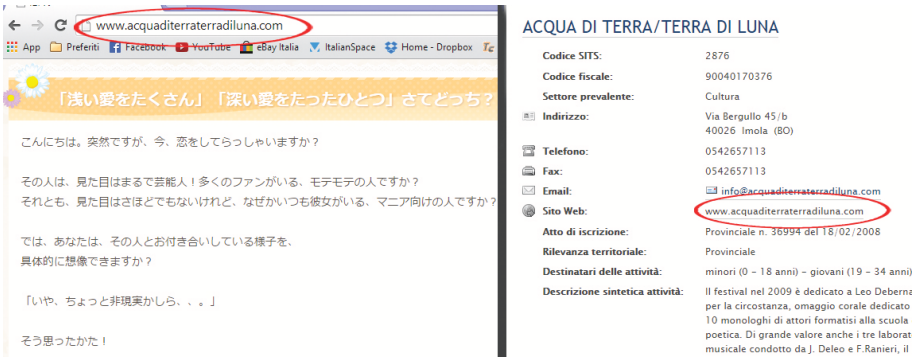

Figura 1.3: Esempio di link che rimanda ad un sito sbagliato

Network. Creare un profilo sui Social è estremamente semplice e alla portata di tutti, per questo sono molte le associazioni presenti su *Facebook*[3] o *Twitter*[4]. Gestire un profilo Facebook, ad esempio, in alcuni casi risulta molto più semplice che gestire un Sito Web e inoltre, grazie agli smartphone e le app dedicate, è possibile aggiornare le informazioni in qualsiasi momento e ovunque ci si trovi. Molte delle associazioni per cui non è indicato un Sito Web di riferimento, risultano invece presenti su Facebook (basta effettuare delle semplice ricerche), questo fa pensare che i Social Network non siano presi in considerazione dagli elenchi citati o semplicemente non sono stati aggiornati recentemente.

Altri problemi sono saltati fuori provando ad effettuare delle ricerche incrociate sugli elenchi e confrontando le informazioni trovate. Si consideri ad esempio il caso riportato in figura 1.4: i dati fanno riferimento all'associazione '*OLTRE...*', ma si nota immediatamente che ci sono delle incongruenze:

- gli indirizzi non corrispondono;
- i link indicati fanno riferimento a due siti web differenti;
- i numeri di telefono e il numero di fax sono diversi.

[3] http://www.facebook.com/
[4] http://twitter.com/

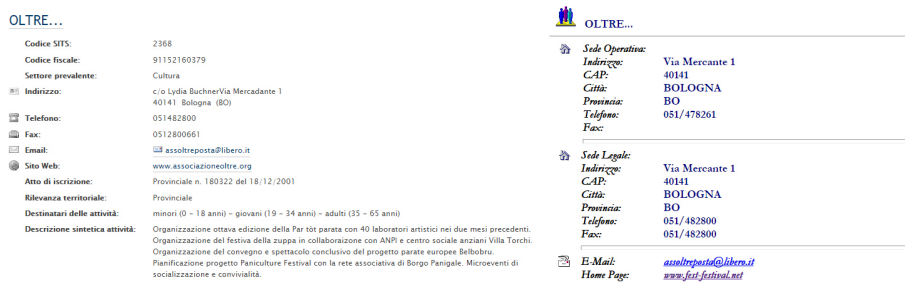

Figura 1.4: Esempio di informazioni non corrispondenti

In casi del genere diventa difficile riuscire a determinare quali siano le informazioni corrette e quali invece quelle errate.

Un altro problema da non sottovalutare riguarda l'eterogeneità dei dati messi a disposizione, causata dal fatto che le informazioni provengono da fonti diverse e sono pubblicate in modo differente; questo costituisce un grosso limite per quanto riguarda la compatibilità e la possibilità di integrazione tra dati provenienti da elenchi diversi.

Mettendo da parte i punti fino ad ora esposti, occorre fare anche un discorso in riferimento a *come* questi dati sono pubblicati. Da un punto di vista tecnico, i formati con cui questi dati sono solitamente messi a disposizione (nei casi considerati si tratta di pagine HTML, ma è frequente anche l'utilizzo di file .PDF o .DOC) non sono *machine-readable*, il che significa che pur essendo liberamente consultabili e comprensibili all'uomo, non possono però essere interpretati e processati automaticamente da delle macchine. Questo rende molto difficile, se non impossibile, utilizzare tali dati come base su cui poter eventualmente creare o fornire dei servizi. In aggiunta, bisogna considerare anche eventuali problemi di utilizzo legati al tipo di licenza con la quale i dati sono rilasciati.

1.2 Obiettivo: un dataset sulle Associazioni Non Profit

Le problematiche introdotte nella sezione precedente hanno portato all'idea di realizzare un sistema che permetta di raccogliere e sistematizzare le informazioni relative alle Associazioni Non Profit presenti sul territorio. Lo scopo è quello di ottenere un dataset curato ed omogeneo, basato su un modello standard, che possa essere utilizzato liberamente da chiunque ne abbia la necessità. Più nello specifico, il fine principale per i nostri interessi, è quello di poter utilizzare i dati raccolti come base su cui modellare un servizio che renda disponibili le informazioni relative agli eventi, corsi e manifestazioni organizzati dalle varie Associazioni.

I dati che ci interessa raccogliere riguardano:

- il nome dell'associazione
- di cosa si occupa
- informazioni riguardo l'ubicazione della sede
- informazioni di contatto (email o numeri di telefono)

1.2.1 L'idea di base

'*Come posso reperire i dati necessari per la composizione del dataset?*' Questa è la prima domanda che mi sono posta.

Quello che mi occorreva era trovare dei mezzi che mi permettessero di raccogliere grandi quantità di informazioni in tempi ragionevoli e nel modo più automatizzato possibile, mantenendo un occhio di riguardo in riferimento alla precisione e alla correttezza dei dati ottenuti. Considerando gli strumenti a mia disposizione ed affrontando il problema dal punto di vista informatico, la soluzione più semplice ed immediata è risultata quella di sfruttare i Siti Web delle Associazioni come fonti dalle quali estrapolare le informazioni ricercate. Al giorno d'oggi infatti, il modo migliore per farsi conoscere ed

essere raggiungibili è quello di essere presenti su Internet ed è per questo che la maggior parte delle Organizzazioni possiede un Sito Web personale, utilizzato per condividere le proprie attività ed avere un contatto diretto e veloce con gli utenti. Per questo motivo, il sito internet di un'Associazione può essere considerato un ottimo punto di partenza da cui poter prelevare le informazioni che ci interessano, con la certezza quasi assoluta della loro correttezza.

Guardando al futuro e pensando all'utilizzo che si vuole fare del dataset che si andrà a creare, l'idea di utilizzare i siti web come principale fonte per il reperimento dei dati risulta ancora più ottimale. Infatti è proprio attraverso questi siti che le associazioni pubblicano le informazioni relative agli eventi e le attività organizzate; effettuando delle analisi periodiche sarà possibile ottenere facilmente dati sempre aggiornati. Un altro aspetto importante da considerare è la possibilità che le associazioni, oltre a possedere un Sito Web, possano avere anche dei profili sui diversi Social Network: nel costruire un tool che permetta di recuperare le informazioni delle associazioni tramite Web occorre tener presente anche queste eventualità.

A questo punto sorgono spontanee altre domande:

1. *come recuperare gli URL dei siti delle Associazioni?*
2. *come estrarre i dati dai siti trovati?*
3. *in che modo sistematizzare e rendere disponibili i dati ottenuti?*

Le risposte ai quesiti appena esposti sono risultate, nell'ordine, le seguenti:

1. utilizzare la tecnica del *Web Crawling* per ricercare all'interno del Web e prelevare gli URL dei siti di interesse;
2. sfruttare il *Web Scraping* per estrapolare le informazioni dai siti trovati;
3. creare e pubblicare il Dataset in un formato semplice, facilmente utilizzabile e che rispetti le specifiche degli *Open Data*.

Le tecniche e i concetti introdotti nei punti appena citati costituiscono una parte fondamentale del progetto svolto e per questo motivo, nelle prossime sezioni, ne verrà fornita una descrizione dettagliata a livello teorico.

1.3 Web Crawling

Un Web Crawler (detto anche Spider o Robot) è un programma software o uno script che naviga all'interno del WWW (World Wide Web) in un modo metodico e automatizzato [2]. I più famosi motori di ricerca (es. *Google*[5], *Yahoo*[6]) utilizzano questi software per trovare le pagine disponibili pubblicamente e organizzare le informazioni ottenute tramite indicizzazione.

Il Crawling del web, può essere visto come una specie di visita su un grafo, in cui le pagine costituiscono i nodi e i link ipertestuali i ponti che li collegano. Il Crawler passa quindi da un sito all'altro, espandendosi come una ragnatela (da qui il nome Spider) sulla rete.

1.3.1 Come lavora un Crawler

Per spiegare il funzionamento base di un Web Crawler occorre introdurre alcuni concetti fondamentali:

Seed Page è il nome con cui viene indicata la lista di URL iniziale a disposizione del Crawler (solitamente fornita dagli stessi programmatori).

Frontier è la coda che contiene gli URL che devono ancora essere analizzati; questa lista viene popolata aggiungendo di volta in volta i link estratti dalle pagine visitate.

Parser si occupa di analizzare le pagine visitate ed estrarre gli URL da aggiungere alla *Frontiera*.

La struttura base di un Web Crawler è mostrata in figura 1.5 [4]

[5]http://www.google.it/
[6]http://it.yahoo.com/

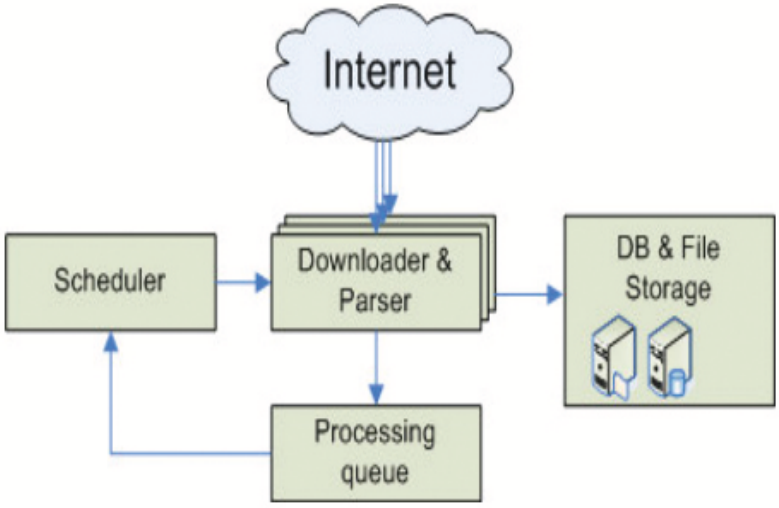

Figura 1.5: Componenti di un Web Crawler

Dal punto di vista logico, il lavoro di un Crawler è molto semplice e può essere schematizzato nel modo seguente:

1. Seleziona uno o più URL dalla Seed Page e li aggiunge alla Frontiera
2. Recupera un URL dalla Frontiera
3. Il Parser analizza la pagina alla ricerca di altri link
4. Aggiunge gli URL trovati alla frontiera
5. Ritorna al punto numero 2 e ripete fino a che la Frontiera non è vuota

1.3.2 Tipologie di Web Crawler

I Web Crawler possono essere suddivisi in diverse tipologie [3], in base al modo in cui lavorano:

General Purpose Crawler Un General Purpose Web Crawler cerca di recuperare il maggior numero di pagine partendo dalla Seed Page ed

espandendosi attraverso tutti i link trovati durante l'analisi. Solitamente la visita delle pagine è effettuata applicando l'algoritmo di visita in larghezza(*Breadth-First*).

Distributed Crawler In questo caso il lavoro del Crawler è suddiviso su più processi. Il numero di pagine esaminate al secondo è indipendente dal numero di processi in esecuzione.

Focused Crawler Si tratta del tipo di Crawler più utilizzato dai motori di ricerca. Lo scopo di un Focused Crawler, infatti, è quello di ricercare e selezionare le pagine che corrispondono ad uno specifico argomento, solitamente definito in base a degli input inseriti dall'utente (o in alcuni casi in base ad un set predefinito dal programmatore).

Il Crawler utilizzato e implementato per il progetto presentato in questo trattato, appartiene all'ultima tipologia elencata (Focused Crawler): lo scopo infatti è quello di riuscire ad ottenere una lista di risultati che appartengono ad uno specifico dominio, ovvero le Associazioni Non Profit.

1.4 Web Scraping

Con il termine Web Scraping ci si riferisce al processo di raccolta delle informazioni contenute all'interno delle pagine Web. Solitamente questo avviene per mezzo di programmi software automatizzati, che simulano la navigazione umana nel World Wide Web.

Più nello specifico, lo scopo principale di un Web Scraper è quello di recuperare e trasformare dati non strutturati o semi-strutturati, in modo che possano essere salvati su un Database secondo una precisa struttura che ne permetta l'utilizzo e li renda machine-readable.

Esistono svariati motivi che possono spingere all'utilizzo di tecniche di scraping: ricerche di mercato, monitoraggio di dati, recupero di informazioni, statistiche; il problema è che gli Scraper non vengono sempre utilizzati con

fini leciti e in alcuni casi vengono sfruttati per rubare i dati personali degli utenti o delle aziende. Il Web Scraping quindi, si trova su una linea sottile che divide il 'recuperare informazioni' dal 'rubare informazioni'. Per questo motivo occorre fare molta attenzione sull'uso che viene fatto degli Scraper ed essere sicuri di non violare i termini di utilizzo dei siti che si vanno ad analizzare.

1.4.1 Funzionamento di un Web Scraper

La logica di base che definisce il funzionamento di un Web Scraper può essere riassunta nei seguenti passaggi:

1. Viene recuperato l'URL del sito che si vuole analizzare
2. Viene effettuata una richiesta HTTP alla pagina
3. Si attende la risposta
4. L'oggetto restituito dalla chiamata viene analizzato per ricercare i dati desiderati
5. I dati ottenuti vengono salvati (su un Database o un file)

I passaggi appena descritti sono schematizzati nella figura 1.6 [5]

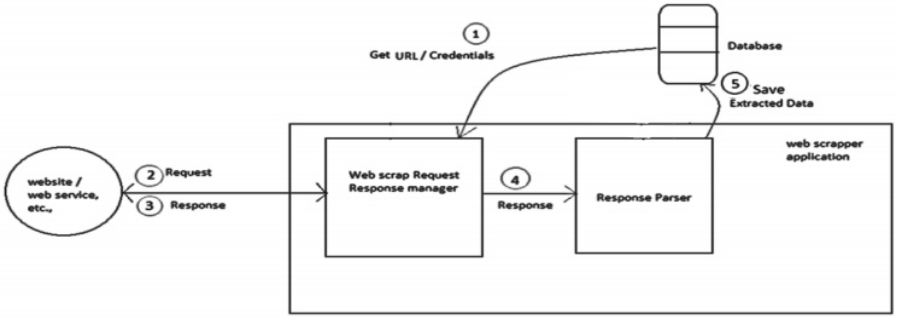

Figura 1.6: Architettura di un Web Scraper

1.4.2 Tecniche di Scraping

Esistono numerose tecniche di Scraping, con le quali è possibile analizzare i siti su diversi livelli e in base alle proprie necessità; le più utilizzate sono le seguenti:

Copy-and-paste si tratta del metodo più semplice, ma anche il più costoso in termini di tempo e precisione. Consiste nel semplice 'copia e incolla' delle informazioni da una fonte all'altra, effettuato manualmente. Nonostante sia il metodo meno utilizzato, in alcuni casi rimane l'unica alternativa possibile.

Regular Expression (regex) Matching si utilizzano le espressioni regolari per ricercare le informazioni in base ad un pattern che esse definiscono. Questa tecnica, applicata singolarmente, può risultare dispendiosa in termini di tempo, in quanto richiede l'analisi approfondita dell'intera pagina; tramite l'utilizzo delle regex però si hanno ottimi risultati per quanto riguarda la precisione e l'accuratezza dei dati recuperati.

HTML/DOM parsing in questo caso viene recuperato il codice HTML associato alla pagina, dal quale vengono estratte le informazioni sfruttando i tag (e relativi attributi) e/o i selettori CSS (Cascading Style Sheets)

Web Scraping software si tratta di programmi che possono essere configurati in modo che possano riconoscere la struttura di una specifica pagina web ed estrapolarne i dati. Esistono centinaia di software di questo tipo, molti dei quali distribuiti con licenza Open Source.

Per recuperare le informazioni necessarie per la composizione del dataset relativo alle Associazioni Non Profit (che rappresenta l'obiettivo di questo lavoro), è stata utilizzata la tecnica di HTML parsing, in combinazione con l'utilizzo delle espressioni regolari.

1.5 Il concetto degli Open Data

Prima di spiegare cosa si intende per *Open Data*, occorre capire da dove questo termine trae origine.

Gli Anni Novanta hanno visto la nascita e lo sviluppo del movimento *Open Source*, che prevede il rilascio libero del codice sorgente dei software, da parte degli autori. Lo scopo dell'Open Source è quello di favorire il libero studio del codice e permettere alla comunità di apportare modifiche, sviluppare migliorie, aggiornamenti o versioni alternative. Queste possibilità sono regolate tramite l'applicazione di apposite licenze d'uso e da una serie di linee guida definite nella *Open Source Definition*[7].

Alla filosofia del movimento Open Source si ispira il movimento Open Data.

Se l'obiettivo primario dell'Open Source è la pubblicazione libera del codice sorgente dei software, nel caso degli Open Data quello che si vuole rendere liberamente disponibile è la 'conoscenza'. Il termine conoscenza racchiude al suo interno contenuti creativi (come libri, musica, video), dati di qualsiasi genere (scientifici, geografici, storici) e informazioni governative e amministrative.

Nel 2004 è nata la *Open Knowledge Foundation*[8] (OKFN), il cui scopo è di promuovere la creazione, la diffusione e l'uso della conoscenza aperta in tutte le sue forme. La OKFN ha pubblicato una serie di punti che definiscono quando un'opera (termine usato per denotare l'oggetto di conscenza) può essere definita aperta, portando così alla creazione della cosiddetta *Open Data Definition*[6]. Gli aspetti fondamentali si possono riassumere in:

Accessibilità L'opera deve essere disponibile nella sua interezza ed a un costo di riproduzione ragionevole, preferibilmente tramite il download gratuito via Internet. L'opera deve inoltre essere disponibile in un formato comodo e modificabile.

[7] http://opensource.org/osd
[8] http://okfn.org/

Licenza aperta L'opera deve essere pubblicata con un tipo di licenza aperta ed eventuali condizioni aggiuntive (ad esempio i termini di utilizzo) non devono contraddire i termini della licenza.

Formati aperti L'opera deve essere distribuita in un formato comodo e modificabile, in modo tale che non vi siano ostacoli o restrizioni sull'utilizzo.

Abbandonando il concetto generale di conoscenza e passando a quello più specifico dei dati, possiamo dire che con il termine Open Data ci si riferisce a dati che possono essere utilizzati liberamente e redistribuiti da chiunque, con l'unico vincolo di condividerli con lo stesso tipo di licenza con la quale sono stati originariamente rilasciati.

Nonostante si parli di un movimento nato più di 10 anni fa, è solamente negli ultimi anni che l'Open Data ha iniziato diffondersi in maniera importante in tutto il mondo. Questa crescita è dovuta principalmente alla nascita di numerose iniziative intraprese dai Governi e dalle Pubbliche Amministrazioni (PA), che hanno annunciato la volontà di rendere pubbliche alcune delle informazioni in loro possesso, con il fine principale di aumentare la trasparenza, la partecipazione e la collaborazione tra le Pubbliche Istituzioni e i cittadini; questo movimento è conosciuto con il nome *Open Government*. Barack Obama, presidente degli Stati Uniti d'America, è stato uno dei primi sostenitori di questa filosofia, come è possibile intuire da [7]. *Data.gov*[9] è il portale statunitense in cui sono pubblicati tutti i dataset prodotti e rilasciati dalle PA locali e rappresenta uno dei primi esempi di diffusione di dati che siano alla portata di tutti. Seguendo l'esempio americano, Nazioni di tutto il mondo hanno intrapreso la strada verso l'apertura dei dati, creando portali ad-hoc per la pubblicazione dei dataset prodotti, uno degli esempi più eclatanti è rappresentato dal portale inglese[10].

Il contributo dato dai Governi risulta fontamentale, non solo per la quantità e centralità dei dati raccolti, ma soprattutto per l'incoraggiamento all'a-

[9] http://www.data.gov/

[10] http://www.data.gov.uk/

Figura 1.7: Diffusione degli Open Data nel mondo[8]

pertura dei dati, fornito alle varie aziende e organizzazioni, e per la diffusione di questa ideologia che altrimenti avrebbe faticato a prendere piede.

L'importanza di aprire i dati non risiede nei soli concetti di conoscenza e trasparenza: ci sono molti contesti in cui i dati aperti possono rappresentare un valore rilevante, come accade nell'ambito della tecnologia e innovazione. Gli Open Data infatti possono rappresentare un ottimo punto di partenza per gli sviluppatori, che possono utilizzarli come base per mettere in opera le proprie idee, creando ad esempio applicazioni o prodotti innovativi che possono dare origine a nuovi business e a nuove start-up.

Uno dei vantaggi principali degli Open Data consiste nella possibilità di combinare tra loro basi di dati differenti, creando un valore aggiunto e generando maggiore informazione. Il concetto di interoperabilità infatti è uno dei principi cardine dell'informazione libera.

Purtroppo, ad oggi, ci sono ancora diverse problematiche e difficoltà legate alla diffusione e all'utilizzo degli Open Data. Le organizzazioni che producono

e raccolgono i dati spesso sono restie a pubblicarli in maniera aperta, perchè credono che 'liberarli' comporti una perdita economica, in quanto significa non poter guadagnare con il lavoro svolto per acquisirli. Per questo motivo finiscono con il porre delle restrizioni sull'utilizzo dei dati pubblicati, andando di fatto contro i principi fondamentali su cui si basa l'ideologia 'Open'. Altri problemi riguardano l'aspetto tecnico e il modo in cui i dataset vengono pubblicati: si parla di dati grezzi e che risultano quindi difficili da interpretare ed utilizzare senza avere delle linee guida da seguire.

Uno dei formati più utilizzati per la pubblicazione di dati aperti è il CSV (Comma Separated Values), in quanto permette la condivisione di grandi quantità di dati in maniera semplice e compatta. Tuttavia, se da un lato la sua semplicità ne rappresenta un punto di forza, dall'altro ne definisce un grosso limite, proprio per le problematiche appena introdotte. Infatti, parlando di file contenenti grandi quantità di dati, risulta difficile riuscire a comprendere il significato delle diverse colonne senza possedere una documentazione relativa alla strutturazione e alla divisione dei diversi campi.

La questione diventa ancora più complessa se si pensa alla possibilità che a interpretare questi dati siano le macchine piuttosto che l'uomo.

Per questo motivo è nata l'esigenza di definire degli standard che permettano un utilizzo più semplice dei dati pubblicati, ma soprattutto che li renda *machine-readable*, ovvero in grado di essere compresi e processati anche da un elaboratore. Quest'asppetto è fortemente collegato all'idea del *Semantic Web*

1.5.1 Cos'è il Semantic Web?

Il Semantic Web è definito come la naturale evoluzione del World Wide Web. Tim Berners-Lee[9], creatore del Web e fondatore del W3C[11], definisce il Semantic Web come *'un'estensione del Web attuale, in cui le informazioni sono strutturate in modo che siano comprensibili ed elaborabili da programmi,*

[11] http://www.w3.org/

aggregando le fonti informative in un'unica entità interrogabile da agenti di ricerca basati su criteri semantici'.

In maniera più semplice, possiamo definire il Semantic Web come il Web in grado di mostrare i propri contenuti in modo che possano essere comprensibili facilmente anche da una macchina e non solo dall'uomo. Per ottenere questo, non viene richiesto ai computer di arrivare a capire il linguaggio umano e la sua logica, ma al contrario deve essere l'uomo a progettare e realizzare in maniera adeguata le risorse Web, in modo da venire in contro alle esigenze delle macchine. L'ostacolo più grande, in questo senso, consiste nel fatto che la maggior parte delle informazioni già presenti sul Web sono progettate esclusivamente per un uso umano e in genere non sono strutturate; è quindi necessario convertire e sistematizzare le risorse attuali (e quelle future), tramite l 'utilizzo di:

- metadati per indicare i collegamenti semantici tra le risorse;
- ontologie per definire uno schema del dominio, in modo da consentire ragionamenti sui collegamenti, effettuare ricerche ed estrarre le informazioni di interesse.

Alcuni degli obiettivi che si pone il Semantic Web sono:

- Permettere l'automazione di operazioni, rendendo l'informazione accessibile in maniera automatica agli agenti software;
- Favorire l'integrazione dei contenuti da sorgenti diverse;
- Rendere efficiente e conveniente la memorizzazione della conoscenza;
- Ottimizzare la ricerca di informazioni mediante i motori di ricerca;
- Garantire un livello di fiducia nella bontà delle risposte ottenute.

Il Web Semantico risulta strutturato su un'architettura a livelli, nella quale sono evidenziate le tecnologie necessarie per la sua realizzazione. Ogni

livello deve essere costruito in modo che rappresenti una base solida e ben definita per i livelli sovrastanti. Diverse tecnologie sono già oggi ampiamente disponibili (Unicode, URI, XML, RDF[12]), altre sono attualmente in fase di sviluppo (OWL), mentre altre rappresentano l'evoluzione futura su cui si sta lavorando (Logica, Prova, Fiducia).

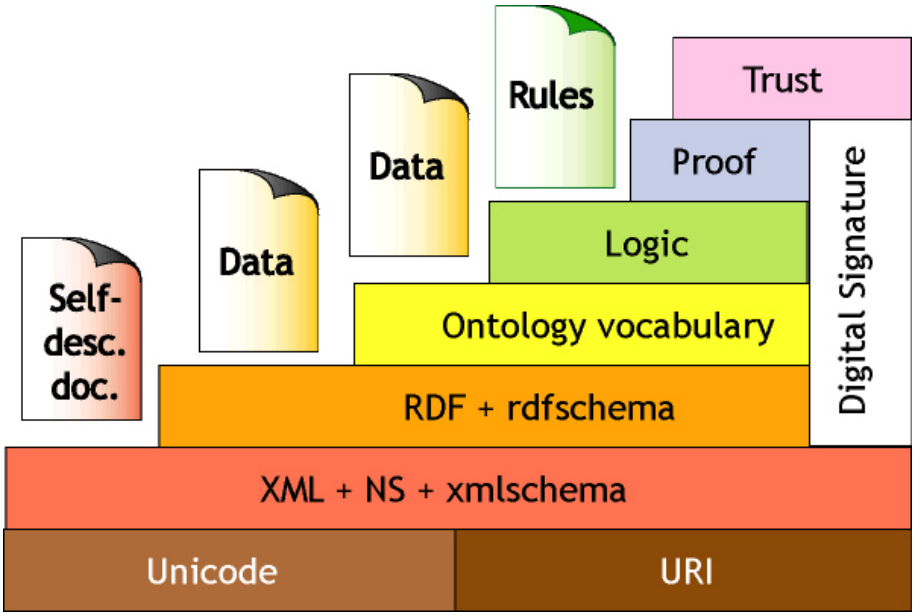

Figura 1.8: Architettura del Semantic Web (Tim Bernsers-Lee)

Tralasciando i livelli più bassi, che rappresentano tecnologie conosciute ed utilizzate ormai da anni, il primo passo importante verso la creazione di un Web Semantico è arrivato con l'introduzione di RDF e le sue successive evoluzioni.

Resource Description Framework (RDF)

Il *Resource Description Framework* è un formato raccomandato dal W3C (World Wide Web Consortium) utilizzato per la codifica e lo scambio di dati strutturati, e che consente una forte interoperabilità tra le applicazioni

[12]http://www.w3.org/RDF/

che condividono le informazioni sul Web. Tramite RDF è possibile introdurre informazioni sui dati stessi, attraverso l'utilizzo dei *metadati*. I metadati sono informazioni strutturate che consentono di descrivere, localizzare e usare più facilmente i dati e gli oggetti, rendendoli maggiormente interpretabili dalle macchine.

Il data model RDF è molto semplice e si basa su tre tipi di oggetti:

risorse qualsiasi oggetto identificabile univocamente mediante un URI (Uniform Resource Identifier).

proprietà un aspetto specifico, una caratteristica o una relazione utilizzata per descrivere una risorsa. Ogni proprietà può essere vista come un tipo speciale di risorsa ed è anch'essa identificata tramite un URI.

statements una tupla composta da un soggetto (risorsa), un predicato (proprietà) e un oggetto (letterale/risorsa); è utilizzata per collegare risorse e le relative proprietà.

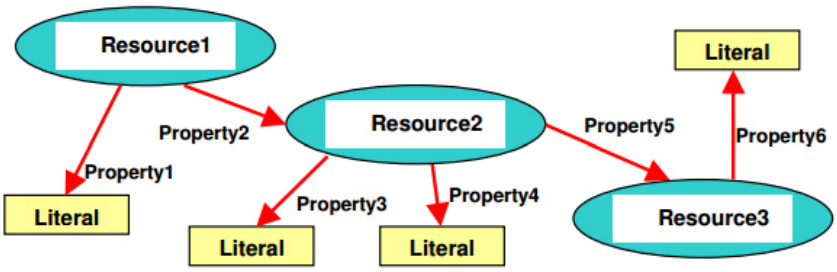

Figura 1.9: Modello generico di una descrizione RDF[10]

Una rappresentazione grafica di una generica descrizione RDF è quella della figura 1.9 .

Riassumendo, RDF fornisce la tecnologia per esprimere il significato di termini e concetti in una forma che i computer possono elabrorare, ma da sola non è sufficiente per riuscire a creare delle strutture semantiche capaci di garantire ragionamenti 'reali' agli agenti software.

Per gli esseri umani risulta molto difficile riuscire ad analizzare una frase privandola del contesto a cui fa riferimento, questo perchè spesso, nel linguaggio naturale, ad un singolo termine possono essere associati più concetti o significati. Lo stesso vale per i computer, che necessitano di sapere qual'è il contesto a cui appartengono i dati che stanno leggendo, in modo da riuscire ad elaborare le informazioni in maniera corretta. Ad esempio, consideriamo la parola 'calcio': essa assume significati diversi a seconda del contesto in cui è utilizzata, può essere associata allo sport ('*Ieri ho ho visto una partita di calcio*'), alla chimica ('*Il calcio è l'elemento chimico di numero atomico 20*') o ancora ad un'azione compiuta ('*Ho tirato un calcio a quel sasso*'); estrapolando la parola 'calcio' al di fuori della frase in cui è inserita, diventa impossibile riuscire a capire quale sia il giusto significato da darle. Occorre quindi trovare un modo per indicare in modo formale il legame tra i termini e i loro significati. Per questo motivo sono stati introdotti *RDF-Schema*[13] (RDFS) e l'*Ontology Web Language*[14] (OWL).

RDF-Schema

Si tratta di un'estensione di RDF che consente di organizzare in tassonomie le classi e le relazioni (proprietà) del dominio, a seconda della loro generalità. Questo è possibile grazie all'introduzione di costrutti a cui sono attribuiti dei significati particolari e che permettono di creare delle Ontologie.

Ad esempio, tramite RDFS è possibile specificare che la classe 'cane' è una sottoclasse della classe 'animale'; questo permette ad una macchina di poter effettuare dei semplici ragionamenti sui dati: se Shila è un cane, allora Shila è un animale. In questo modo si riescono a ricavare delle informazioni che non erano esplicitamente indicate nei dati iniziali. Il processo di ragionamento ('Reasoning') appena introdotto rappresenta uno dei cardini del Semantic Web.

[13] http://www.w3.org/TR/rdf-schema/
[14] http://www.w3.org/TR/rdf-schema/

Anche questo però non è sufficiente per riuscire ad ottenere una struttura solida e ben definita: RDFS infatti si limita a definire le relazioni di ereditarietà, ma non specifica nulla riguardo la cardinalità e l'utilizzo delle proprietà definite.

Ontology Web Language (OWL)

Tom Gruber ha definito l'ontologia come *'una specifica formale, di una concettualizzazione condivisa'* [15]. Le ontologie sono molto importanti, in quanto rappresentano uno dei pilastri del Semantic Web.

L' OWL è un linguaggio definito con lo scopo di arricchire le specifiche dei concetti RDFS. In particolare, OWL introduce costrutti per la definizione delle cardinalità, delle relazioni tra le classi e definisce i concetti di simmetria, transitività per le proprietà.

Nel contesto del Semantic Web, sono state definite diverse ontologie che costituiscono uno standard per la descrizione di svariate risorse. Nel contesto di questo lavoro di tesi, l'ontologia di riferimento è l'*Organization Ontology*[15]; si tratta di un'ontologia definita dal W3C, che risulta fondamentale per la descrizione di strutture organizzative ed è progettata per consentire estensioni specifiche del dominio, come ad esempio aggiungere la classificazione delle organizzazioni, le attività svolte, i ruoli interni e le strutture.

Altre ontologie che vale la pena di citare, in quanto collegate all'Organization Ontology sono *FOAF*[16], *vcard*[17] e *SKOS*[18]:

FOAF (Friend Of A Friend) descrive le persone, le loro attività e le relazioni che possono avere con altre persone o altri oggetti.

[15]http://www.w3.org/TR/vocab-org/
[16]http://xmlns.com/foaf/spec/
[17]http://www.w3.org/TR/vcard-rdf/
[18]http://www.w3.org/TR/skos-reference/

vcard specifica sviluppata per la descrizione di persone ed organizzazioni; in particolare permette di definire informazioni riguardo il luogo e i contatti delle entità indicate.

SKOS (Simple Knowledge Organization System) è un modello che serve per rappresentare glossari, clssificazioni, tassonomie.

Maggiori dettagli riguardo la struttura e l'uso che è stato fatto delle varie ontologie sono forniti nel capitolo 2, sezione 2.1.

1.5.2 Il progetto 'Open Linked Data'

La strada verso la realizzazione di un Web Semantico risulta però ancora lunga. Negli ultimi anni si è cercato di adattare i dati già presenti sul Web per poter passare dal cosiddetto *Web of documents*, la cui unità principale è costituita dai documenti HTML, al *Web of data*, costruito invece, attorno al concetto principale di *risorsa*. Il risultato è stato quello di ottenere tanti insiemi di risorse che risultano ben organizzati e connessi al loro interno (tramite statement RDF), ma allo stesso tempo fortemente scollegati tra loro esternamente. Per questo è nata la necessità di trovare un modo per poter collegare tra loro questi insiemi e mettere in relazione risorse provenienti da fonti diverse, ed è proprio questo l 'obiettivo dei Linked Data[19].

Con il termine Linked Data ci si riferisce ad un insieme di pratiche per la pubblicazione e il collegamento di dati strutturati sul Web. L'obiettivo dei Linked Data è quello di riuscire ad esprimere i significati delle informazioni presenti sulla rete e renderle condivisibili tra le differenti applicazioni in modo semplice e funzionale.

Berners-Lee [11], nel 2006, ha definito quelli che sono conosciuti come i 'Principi del Linked Data':

- usare gli URI per identificare gli oggetti;

[19]http://linkeddata.org/

- usare URI HTTP in modo che questi oggetti possano essere cercati da persone ed applicazioni web;
- fornire informazioni utili sugli oggetti cercati, utilizzando formati standard (RDF, OWL);
- includere collegamenti ad altri oggetti (sempre secondo lo stesso meccanismo), per aumentare e migliorare il reperimento di altre informazioni correlate.

L'Open Linked Data è un progetto ideato dal W3C che prevede la realizzazione di dataset che siano:

- prodotti seguendo le specifiche dei Linked Data;
- pubblicati secondo i principi degli Open Data.

L'obiettivo è quello di estendere il Web attraverso la pubblicazione di diversi set di dati RDF, che siano connessi tra loro ed utilizzabili in maniera libera e aperta. I tipi di dati pubblicati seguendo queste direttive presentano infatti alta interoperabilità, riusabilità e sono estremamente compresibili sia dall'uomo che dalle macchine.

1.5.3 Stato attuale degli Open Linked Data in Italia

Il progetto LOD (Linked Open Data) è inizialmente partito con la collaborazione di piccoli enti e ricercatori, ed ha ottenuto nel tempo numerose adesioni da parte di istituzioni autorevoli e di grandi dimensioni, comprese alcune unità governative. Questo ha portato ad una notevole espansione del progetto, che sta via via prendendo largo in tutto il mondo.

E in Italia?

Il movimento LOD è nato in Italia solo da pochi anni e per questo si trova ancora in una fase poco avanzata. A Ottobre 2014, nella nostra nazione

risultano disponibili più di 12000 dataset pubblicati in formato aperto [12] ed accessibili attraverso il portale dati.gov.it[20]. Il grafico proposto in figura 1.10 riporta la ripartizione dei dataset in base alla Scala di Tim Berners-Lee [11]; è possibile notare l'evoluzione che c'è stata sia per quanto riguarda il numero di dataset pubblicati, sia per quanto riguarda la qualità ad essi associata.

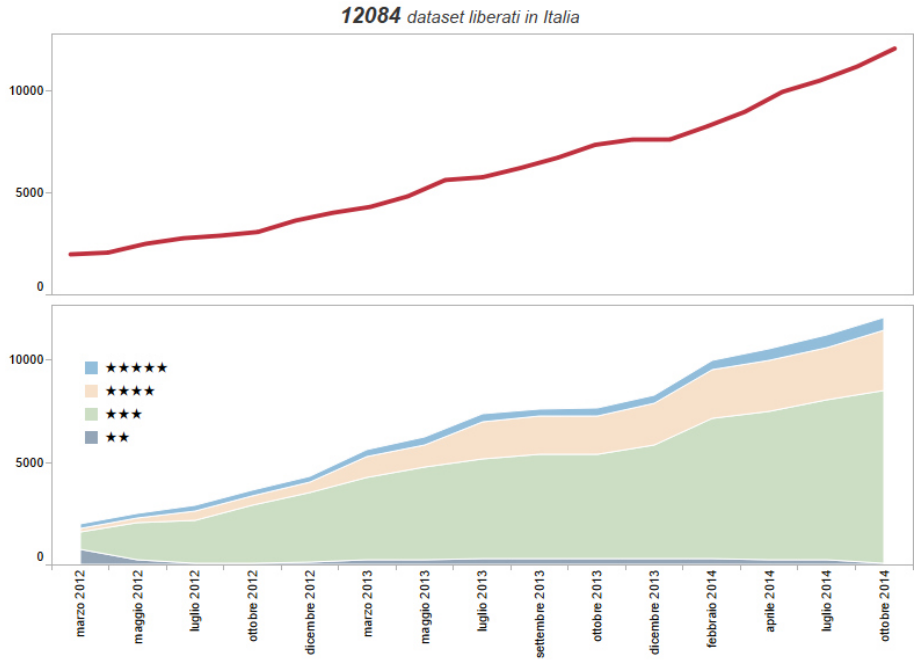

Figura 1.10: Ripartizione dei dataset Italiani (2014)

Occorre tenere presente alcune considerazioni: la classificazione proposta da Berners-Lee riserva un punteggio di 3 stelle per i dati pubblicati in maniera aperta (CSV, XML, JSON,..), mentre punteggi superiori vengono assegnati ai dataset che utilizzano standard internazionali (RDF), tipici dei Linked Data. Ritornando al grafico quindi, se da un lato sono stati fatti grossi passi in avanti per quanto riguarda l'apertura dei dati, dall'altra ne servono ancora tanti in direzione degli Open Linked Data.

[20]http://www.dati.gov.it/

Linked Open Data nel comune di Bologna

Il progetto *OpenData del Comune di Bologna*[21] fa parte del percorso partecipativo dell'*Agenda Digitale*[22] e coinvolge, a livello interdipartimentale, tutti gli enti pubblici del Comune.

Il progetto è stato sviluppato su più fasi ed è iniziato con la realizzazione di un censimento del portafoglio applicativo degli enti, in cui sono stati coinvolti tutti i Settori dell'Amministrazione. L'indagine è iniziata a Maggio 2011 ed è terminata nel Settembre dello stesso anno ed ha portato all'analisi complessiva di 135 applicazioni utilizzate all'interno di ciascun Settore/Area Comunale. I risultati ottenuti sono stati, nell'ordine, i seguenti:

1. Realizzazione di una banca dati delle applicazioni.
2. Analisi di alcune caratteristiche chiave di ciascuna applicazione.
3. Rilevazione dei livelli di soddisfazione degli utenti delle applicazioni
4. Individuazione dei potenziali di Open Data e calcolo di un Open Data Index

L'*Open Data Index* è un indice creato con lo scopo di misurare la propensione alla pubblicazione dei dati da parte di un'applicazione. L'indice, che può variare da 0 a 100 punti, prende in considerazione cinque parametri:

1. il livello di open data già raggiunto e i suoi potenziali per cittadini, imprese e istituzioni.
2. la facilità di esportazione delle informazioni.
3. le tipologie di formati in cui si mettono a disposizione i dati.
4. la qualità dei dati disponibili.
5. la tempestività dei dati disponibili in relazione alle finalità d'uso.

[21] http://dati.comune.bologna.it/
[22] http://www.agid.gov.it/

Sul portale è possibile consultare una versione interattiva dell'indice prodotto [14].

Il progetto Open Data del Comune di Bologna ha portato dal 2011 ad oggi, alla raccolta e alla pubblicazione online di circa 240 dataset in formato aperto. I dati pubblicati riguardano il territorio (quartieri, strade, aree di circolazione), luoghi e personaggi di interesse turistico, edifici pubblici, statistiche. Per quanto riguarda il formato con cui questi dati sono messi a disposizione, il più utilizzato risulta il CSV (82%).

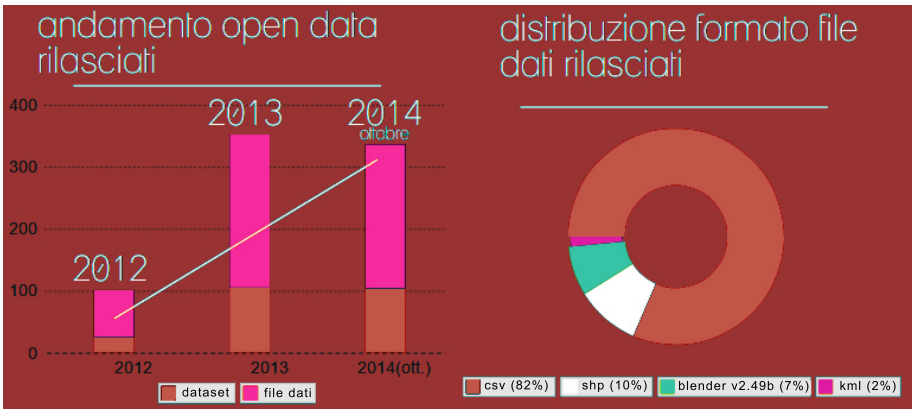

Figura 1.11: Open Data del comune di Bologna (statistiche)

Nell'Ottobre 2014 è stato effettuato un ulteriore passo in avanti, ed è stata rilasciata la versione beta del portale 'Linked Open Data del comune di Bologna', raggiungibile al link: http://linkeddata.comune.bologna.it/ . Il progetto *Linked Open Data* ha portato alla definizione di un vocabolario ad-hoc, denominato *Onto Municipality*, utilizzato per definire le risorse e le loro relazioni; lo scopo era quello di definire una struttura che potesse soddisfare le esigenze di una Pubblica Amministrazione locale, come ad esempio un Comune.

A Gennaio 2015 i dati disponibili risultano ancora pochi; si contano infatti solamente 8 dataset, che riguardano principalmente informazioni geografiche

relative alla città di Bologna (suddivisione dei guartieri, zone, numerazione civica, strade, aree di circolazione,..). Si tratta, comunque, di un progetto avviato recentemente e che si trova attualmente ancora in una fase beta, da un lato quindi non c'è da stupirsi se i numeri risultano ancora molto bassi.

I dati sono messi a disposizione per mezzo di un triple store *Virtuoso Open Source*, tramite il quale è possibile effettuare delle interrogazioni con query SPARQL o semplicemente consultare i dati disponibili navigando sul portale.

1.6 Open Civic Data

Uno progetto interessante e molto vicino al contesto di questo trattato è *Open Civic Data*[23]. Si tratta di un progetto nato per definire delle linee guida per la pubblicazione di informazioni riguardanti alcuni dei soggetti più comuni nell'ambito dei Civic Data. Definendo degli standard sulla pubblicazione di queste informazioni, è possibile aumentare notevolmente le possibilità legate al loro utilizzo. Inoltre, avere dati simili formattati tutti secondo uno schema comune, permette di effettuare confronti tra dataset relativi a diverse città, regioni o nazioni, ad esempio per fini statistici.

Uno dei punti su cui si concentra maggiormente il progetto Open Civic Data, è quello di fornire un formato comune di identificatori univoci per le entità politiche a livello mondiale. Ogni oggetto, ogni risorsa deve avere un ID che la contraddistingue e che deve essere utilizzato ogni qual volta si vogliano esprimere delle informazioni aggiuntive in riferimento alla risorsa considerata.

Lo scopo finale, legato al progetto, è quello di riuscire a costruire un sistema in grado di essere utilizzato in tutto il mondo, e che favorisca il riuso di ciò che viene creato; l'idea è che se un gruppo di persone sviluppa un'applicazione che aiuta i cittadini a monitorare tutte le attività del consiglio scolastico locale, la stessa applicazione potrà essere usata ugualmente anche

[23]http://opencivicdata.org/

in altre città, fino a quando i dati di base sono disponibili e strutturati secondo il formato prescritto da Open Civic Data.

Come si legge sul Sito Web del progetto, per raggiungere gli obiettivi prefissati Open Civic Data necessità della collaborazione di tutti, dai cittadini al Governo. In uno scenario ideale, i Governi adotteranno gli standard definiti per la costruzione e la pubblicazione dei loro dataset, per permettere collegamenti affidabili tra i dati delle varie città.

Persone e Organizzazioni

Le Organizzazioni e le Persone rappresentano due delle entità principali definite da Open Civic Data. Nel contesto specifico di riferimento, le definizioni associate a queste entità sono le seguenti:

Persona Un individuo che ha prestato servizio in un ufficio politico.

Organizzazione Un gruppo di persone che opera in un contesto legislativo o normativo.

Per definire i tipi appena descritti, Open Civic Data ha adottato (con differenze minime) gli schemi proposti da *Popolo*. Popolo è un progetto il cui obiettivo è quello di definire formati di interscambio di dati e modelli di dati per le organizzazioni politiche.

Le strutture definite da Popolo (e quindi utilizzate anche da Open Civic Data) si basano sulle specifiche prescritte dal W3C. In particolare, per quanto riguarda la definizione e la descrizione delle organizzazioni viene fatto riferimento all'Organization Ontology (ORG).

Oltre a definire gli standard per la rappresentazione delle entità considerate, Open Civic Data mette a disposizione gli strumenti necessari per effettuare delle operazioni di scraping utili per reperire i dati di interesse.

Capitolo 2

Il progetto OnDa: cos'è e come funziona

2.1 Scopo del progetto

Gli obiettivi definiti dal progetto sono due: il primo consiste nell'implementazione di un tool in grado di recuperare le informazioni sulle Associazioni Non Profit presenti sul Web; il secondo obiettivo consiste nell'utilizzare le informazioni raccolte per la creazione di un dataset che rispetti le specifiche dei Linked Open Data.

In particolare, per questo lavoro di tesi e la creazione del dataset, è stato preso in considerazione il mondo del Non Profit a livello della provincia di Bologna, con una maggiore concentrazione sulla presenza su Web delle associazioni. Fatto presente questo, la realizzazione è stata comunque pensata in modo da poter essere adattata a diverse esigenze e poter essere in futuro utilizzata in riferimento ad altre province, o ugualmente anche a livello regionale o nazionale. Un altro occhio di riguardo è stato dato alla possibilità di integrare il dataset con informazioni e dati provenienti da altre fonti. Per questo motivo si è scelto di mantenere una certa separazione tra le varie componenti del tool implementato, che risulta quindi suddiviso in diversi moduli, ognuno dei quali con una specifica funzione.

Il funzionamento del tool può essere riassunto nei seguenti passaggi:

Figura 2.1: Funzionamento del tool

1. Viene effettuata una ricerca per recuperare gli URL dei siti web relativi alle Associazioni Non Profit; gli URL trovati vengono inseriti in un elenco memorizzato su un database.

2. I siti presenti in elenco vengono analizzati per cercare di estrapolare le informazioni ricercate; i dati raccolti sono salvati sul database.

3. I dati presenti sul database sono convertiti in triple RDF e memorizzati su un file.

4. Vengono resi pubblici i dati mettendoli a disposizione attraverso uno SPARQL endpoint.

Per favorire la consultazione del materiale raccolto, è stata creata anche un'applicazione web tramite la quale è possibile leggere e ricercare all'interno del dataset le informazioni di interesse.

2.2 Struttura dei dati

Il dataset è creato e pubblicato in formato RDF, per seguire le specifiche imposte dall'Open Linked Data Project.

Per rappresentare le associazioni è stato utilizzato il vocabolario *Organization Ontology*. Questa scelta è stata fatta in quanto si tratta di uno standard raccomandato dal W3C che permette di descrivere le organizzazioni dal punto di vista strutturale e logistico.

Il namespace di riferimento dell' ontologia è *http://www.w3.org/ns/org#* . Per maggiore chiarezza e comprensione degli esempi riportati in questa sezione, di seguito sono elencati tutti i namespace e i prefissi utilizzati (tutti gli esempi sono scritti utilizzando il linguaggio *turtle* (N3)).

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix org: <http://www.w3.org/ns/org#>.
@prefix vcard: <http://www.w3.org/2006/vcard/ns#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix onda: <http://www.progetto_onda.it/>.
```

2.2.1 L'Organization Ontology

La classe principale è *org:Organization*, che viene utilizzata per rappresentare un insieme di persone organizzate in una comunità, che hanno un

fine comune ed operano insieme per raggiungere uno scopo. La classe Organization ha a disposizione una serie di proprietà, con le quali è possibile specificare le informazioni relative all'organizzazione e definirne la struttura gerarchica (unità, sottounità,...), i membri, la sede (località, contatti,...).

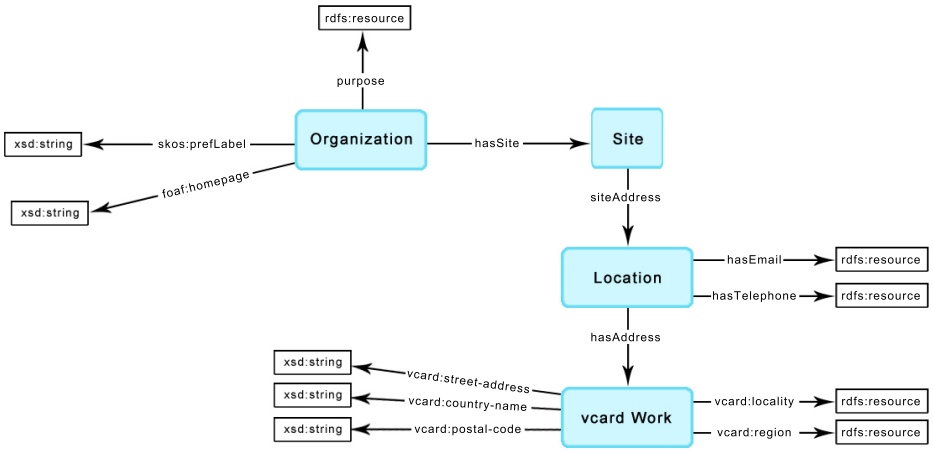

Figura 2.2: Sezione dell'ontologia Organization

Per capire meglio come è strutturato il dataset, vediamo un esempio relativo alla rappresentazione di un'associazione qualsiasi:

```
onda:riunite a org:Organization;
skos:prefLabel "Associazioni riunite" ;
foaf:homepage "http://riunite.eu/" ;
org:purpose onda:cat-Lavoro,
       org:purpose onda:cat-Tutela_dei_cittadini ,
       onda_cat-Associazione_non_profit ;
org:hasSite onda:site-riunite_1 .
```

skos:prefLabel e foaf:homepage

Tramite le proprietà *skos:prefLabel* e *foaf:homepage* vengono indicati rispettivamente il nome dell'associazione e l'URL del relativo sito web.

org:purpose

La proprietà *org:purpose* serve in generale per indicare lo scopo dell'organizzazione (sono ammessi valori multipli); nel nostro caso è utilizzata per indicare la categoria (o le categorie) a cui l'associazione appartiene.

Ogni categoria è rappresentata da un oggetto di tipo *skos:Concept*, descritto nel modo seguente:

```
onda_cat-Associazione_non_profit a skos:Concept;
       rdfs:label "Associazione Non Profit".
```

Tramite la proprietà *rdfs:label* viene fornita una descrizione della categoria in un formato comprensibile all'utente.

org:hasSite

La proprietà *org:hasSite* serve a specificare la sede di un'organizzazione. La sede deve essere dichiarata con un'istanza di *org:Site* alla quale può essere associato un indirizzo, attraverso l'utilizzo della proprietà *org:siteAddress*. Un indirizzo è rappresentato da un oggetto di tipo *vcard:Location* che permette, inoltre, di specificare le informazioni di contatto (email e numeri di telefono).

```
onda:site-riunite_1 a org:Site ;
org:siteAddress onda:loc-riunite_1 .

onda:loc-riunite_1 a vcardLocation ;
vcard:hasEmail <mailto:info@riunite.eu>,
  <mailto:cittadinanzattivanavile@comunitasociale.it>;
vcard:hasTelephone onda:number-0516353605 ;
vcard:hasAddress onda:via_marco_polo_51_40131 .
```

Con le proprietà *vcard:hasEmail* e *vcard:hasTelephone* è possibile specificare, nell'ordine, i contatti email e i contatti telefonici associati alla sede descritta (sono ammessi valori multipli). Ogni numero di telefono è rappresentato da un oggetto di tipo *vcard:Voice*, nel modo seguente:

```
onda:number-0516353605 a vcard:Voice ;
 vcard:hasValue <tel:0516353605> .
```

La proprietà *vcard:hasAddress* serve per indicare l'indirizzo fisico in cui ha sede l'associazione; questo avviene attraverso l'utilizzo di un'istanza di *vcard:Work*.

```
onda:via_marco_polo_51_40131 a vcard:Work ;
 vcard:country-name: "Italia" ;
 vcard:region: "BO" ;
 vcard:postal-code: "40131" ;
 vcard:locality: "Bologna" ;
 vcard:street-address: "Via Marco Polo 51" .
```

Le proprietà specificano, nell'ordine: nazione, provincia, codice di avviamento postale (CAP), comune, via.

In figura 2.3 è riportato uno schema grafico che rappresenta i collegamenti tra le triple descritte nei passaggi precedenti, relativamente all'esempio.

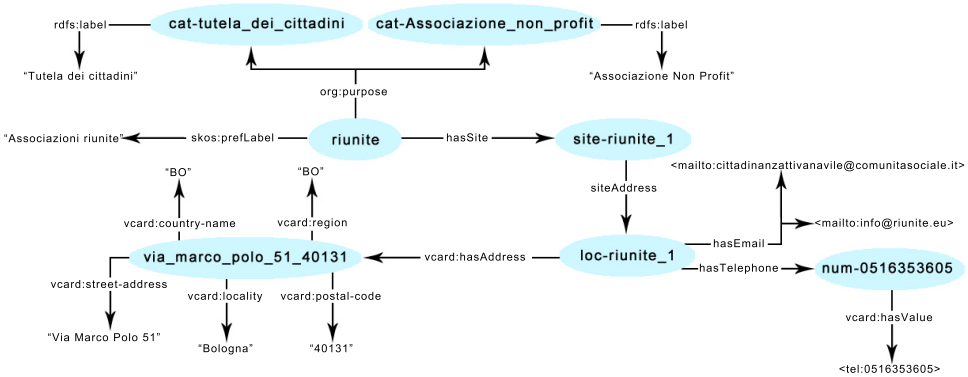

Figura 2.3: Schema grafico dell'esempio citato

2.3 Leggere i dati

Il dataset prodotto può essere consultato in due modi:

- accedendo direttamente allo SPARQL End-Point;
- utilizzando l'applicazione Web messa a disposizione.

2.3.1 Interrogare lo SPARQL End-Point

Lo SPARQL End-Point è raggiungibile al seguente indirizzo:
`http://130.136.131.112:3030`

Per ottenere l'elenco completo delle associazione, con tutti i relativi dati si può utilizzare la seguente query:

```
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix org: <http://www.w3.org/ns/org#>
prefix vcard: <http://www.w3.org/2006/vcard/ns#>
prefix foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?nome_associazione ?link ?stato ?regione ?cap
       ?locality ?indirizzo
(GROUP_CONCAT(DISTINCT ?email ; separator= ';') AS
 ?contatti_email)
(GROUP_CONCAT(DISTINCT ?numero ; separator= ';') AS
 ?contatti_tel)
(GROUP_CONCAT(DISTINCT ?purp_label ; separator= ';') AS
 ?categorie)
WHERE{
?org a org:Organization;
skos:prefLabel ?nome_associazione;
org:purpose ?purpose.
?purpose rdfs:label ?purp_label.
OPTIONAL{ ?org foaf:homepage ?link}
```

```
 OPTIONAL{ ?org org:hasSite ?site.
        ?site a org:Site;
        org:siteAddress ?site_address.
        ?site_address a vcard:Location.
        OPTIONAL {?site_address vcard:hasAddress ?
           address.
         ?address a vcard:Work.
         ?address vcard:country-name ?stato;
         vcard:region ?regione;
         vcard:locality ?locality.
         OPTIONAL{?address vcard:postal-code ?cap}
         OPTIONAL{?address vcard:street-address ?
            indirizzo.}
        }
        OPTIONAL {?site_address vcard:hasEmail ?email}
        OPTIONAL {?site_address vcard:hasTelephone ?
           telephone.
         ?telephone a vcard:Voice;
         vcard:hasValue ?numero
        }
}}
 GROUP BY ?nome_associazione ?link ?stato ?regione ?cap
 ?locality ?indirizzo ORDER BY ?nome_associazione
```

Altre query utili per poter consultare i dati messi a disposizione, sono elencate nell'appendice A.

2.3.2 Consultare i dati tramite Web

Per rendere facilmente consultabili i dati anche per gli utenti che non hanno conoscenze riguardo l'utilizzo di SPARQL o semplicemente non sono interessati alla strutturazione interna del dataset, ho creato l'interfaccia Web denominata *OnDa Viewer*, raggiungibile al seguente indirizzo:

`http://130.136.131.112`

La pagina risulta divisa in una serie di pannelli.

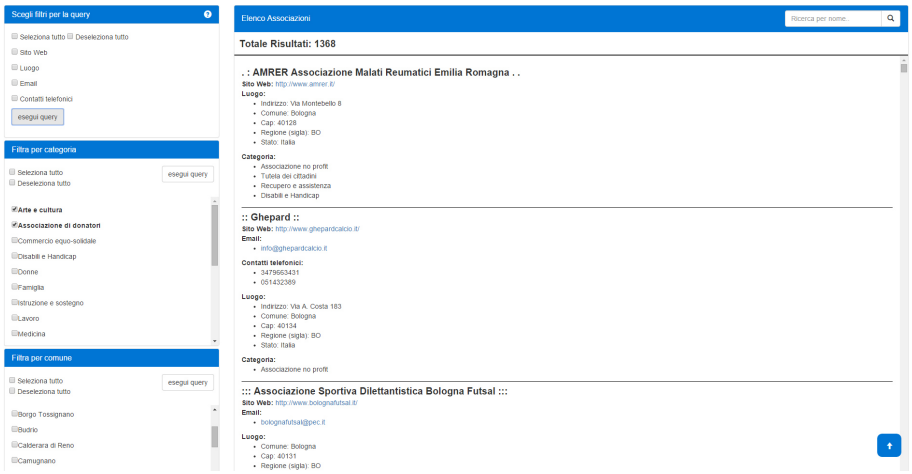

Figura 2.4: Schermata della Web Api

Il pannello principale, denominato *Elenco Associazioni*, contiene l'elenco completo delle associazioni presenti sul Dataset. L'elenco è visualizzato in ordine alfabetico crescente e contiene tutti i dati relativi alle associazioni Nell'header del pannello è presente una barra di ricerca che permette di filtrare l'elenco dei risultati in base al nome delle associazioni.

I pannelli posti sulla sinistra della pagina invece, permettono di effettuare una serie di richieste, per filtrare l'elenco in base a dei parametri selezionati.

La prima tipologia di filtri applicabili, riguarda i dati associati a ciascuna organizzazione e permette di selezionare i campi che si ritengono obbligatori: in questo modo saranno visualizzate nell'elenco solamente le associazioni per le quali si conoscono i dati richiesti. Per esempio, selezionando l'opzione *'Email'*, sarà visualizzato l'elenco delle associazioni a cui risultano associati uno o più contatti di posta elettronica.

Il secondo pannello dei filtri, permette di effettuare una ricerca in base ad una o più categorie selezionate: in questo modo saranno visualizzate nell'elenco solamente le associazioni che appartengono alle categorie di interesse. Per esempio, selezionando le opzioni *'Musica'* e *'Arte e cultura'*,

sarà visualizzato l'elenco delle sole associazioni che rientrano in queste due categorie.

Il terzo e ultimo pannello dei filtri, permette di effettuare una ricerca in base ad uno o più comuni selezionati: in questo modo saranno visualizzate nell'elenco solamente le associazioni che hanno la propria sede nei comuni di interesse. Per esempio, selezionado l'opzione *'Casalecchio di Reno'*, sarà visualizzato l'elenco delle sole associazioni che hanno sede in questo comune.

Il pannello *Statistiche* contiene dei grafici che permettono di avere una stima dei dati raccolti.

Grafico categorie grafico a torta che rappresenta la ripartizione delle associazioni rispetto alle categorie.

Grafico dati grafico a colonne che mostra il numero di dati raccolti suddivisi per il tipo di informazione che forniscono.

2.4 Raccogliere i dati: funzionamento dei tools

La realizzazione del dataset è stata possibile grazie all'utilizzo di due tools:

- il Crawler, che si occupa di recuperare gli URL dei siti web delle associazioni;
- lo Scraper, tramite il quale vengono recuperate le informazioni relative alle associazioni;

Di seguito viene descritto il funzionamento e l'utilizzo dei due componenti appena citati. Una descrizione più approfondita riguardo la loro struttura e l'implementazione è fornita nel capitolo 3.

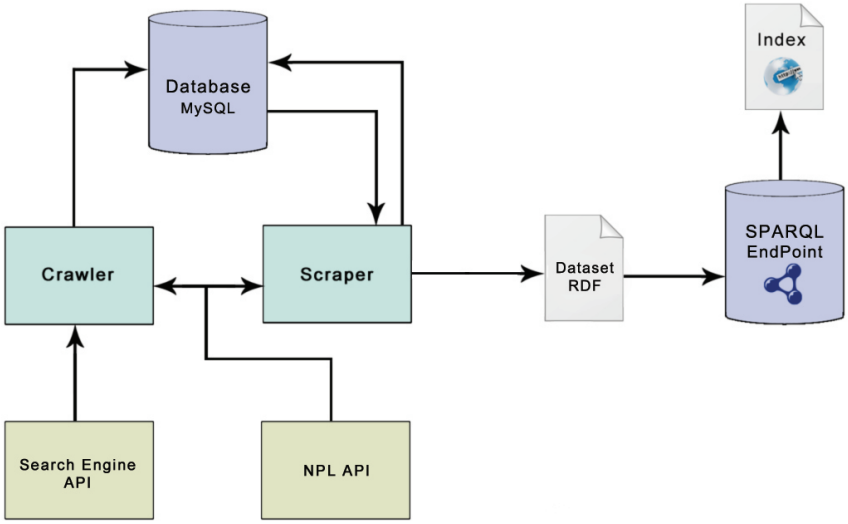

Figura 2.5: Struttura dei tools

2.4.1 Il Crawler: ricerca dei siti

Il Crawler serve per cercare e recuperare i siti internet delle associazioni presenti sul Web. Lo scopo di ogni esecuzione è quello di arricchire e aumentare l'elenco dei siti delle associazioni che andranno a comporre il dataset finale.

Lo script può essere eseguito in due modi:

- Tramite interfaccia web
- Direttamente da linea di comando

Si può accedere all'interfaccia Web utilizzando un normale Browser, digitanto l'indirizzo /crawler.php. Tramite l'utilizzo di una barra di ricerca personalizzata, è possibile inserire i termini o le parole chiave relative alla ricerca che si vuole effettuare; inoltre è possibile applicare dei filtri alla ricerca, in base alla selezione di una o più regioni, o di una o più province. In questo caso è possibile effettuare solo una singola ricerca alla volta e ogni ricerca successiva richiederà una nuova esecuzione.

Per lanciare lo script da linea di comando occorre specificare come parametro dell'esecuzione i termini sui cui si vuole effettuare la ricerca, compresi eventuali filtri riguardanti la località. In questo caso non vi sono limiti per quanto riguarda il numero di ricerche da poter effettuare con una singola esecuzione. È possibile infatti effettuare più ricerche distinte, racchiudendo i termini di ogni ricerca tra virgolette e separandoli da uno spazio: in questo modo verranno interpretati come parametri distinti, per ognuno dei quali sarà effettuata una ricerca.

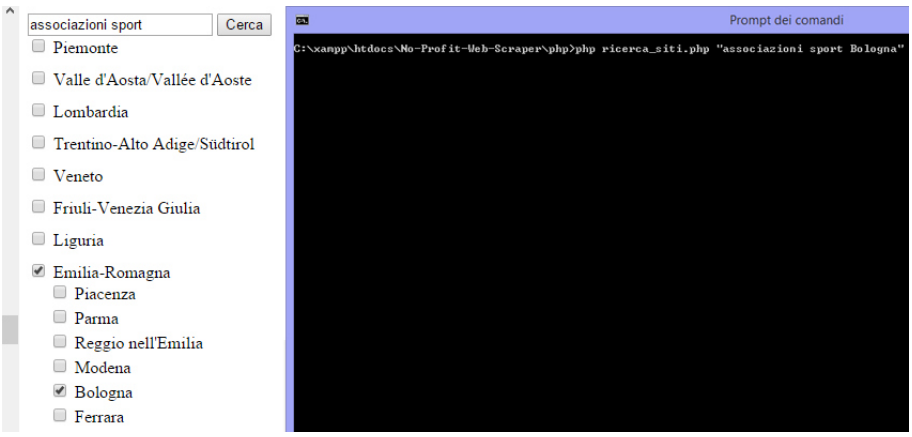

Figura 2.6: Esempio di esecuzione per entrambe le tipologie

Il risultato di ogni ricerca effettuata, consiste in una lista di URL che devono essere analizzati. Già durante questa fase viene effettuata una prima scrematura e vengono scartati i siti che non risultano attinenti alla ricerca effettuata o che non riguardano Associazioni o Organizzazioni Non Profit. Viene inoltre effettuato un confronto con i siti già presenti in elenco, in modo da evitare risultati doppi e ottenere così un dataset più curato.

Al termine dell'esecuzione vengono mostrati a video i risultati ottenuti e i siti trovati. In particolare per ogni URL restituito dalla ricerca, viene indicato se il sito è stato scartato o se invece è stato salvato. Questi risultati vengono anche inseriti nel file di Log prodotto dallo script, un esempio è il seguente:

```
Log generato in data: 25-01-2015 08:01
Termine di input: associazioni sport bologna
Sito scartato: http://bo.paginegialle.it
Sito scartato: http://www.bridgebologna.it
Sito scartato: http://www.sportellodellosport.it
Sito scartato: http://www.acquadela.it
Sito inserito: http://www.as2000.it
Sito inserito: http://www.ascozola.net
Sito inserito: http://www.pgmbologna.com
Sito inserito: http://www.uszinellacsi.it
Sito inserito: http://www.scuoladisportcinquecerchi.it
...
Numero di risultati ottenuti: 40
Numero di risultati scartati: 25
Numero di risultati salvati: 15
```

2.4.2 Lo Scraper: recupero delle informazioni

Grazie al Crawler descritto nella sezione precedente è possibile ottenere un elenco di siti web relativi a delle Associazioni. Il passo successivo consiste nell'analizzare i siti trovati, con il fine di reperire le informazioni di interesse. Questo compito viene eseguito dallo Scraper.

Anche in questo caso, lo script di analisi può essere eseguito in due modi differenti:

- Tramite interfaccia web all'indirizzo /script_ricerca_dati.php
- Direttamente da linea di comando, semplicemente digitando il comando php script_ricerca_dati.php

Non ci sono particolari differenze relative ai due diversi metodi di esecuzione, lo script non richiede parametri in ingresso e una volta lanciato effettua tutto in modo automatizzato. Il funzionamento dello script può essere riassunto brevemente nei seguenti passaggi:

1. Recupero degli URL dei siti da analizzare;
2. Analisi di ciascun sito e recupero delle informazioni;
3. Salvataggio dei risultati;
4. Produzione del dataset in diversi formati;

L'esecuzione non prevede un'analisi totale di tutti gli URL presenti nell'elenco, ma vengono presi in considerazione solamente i siti che non sono mai stati analizzati o per cui è trascorso più di un mese dall'ultima analisi.

Le informazioni recuperate per ciascun sito riguardano:

- Nome dell'associazione
- Categoria di riferimento
- Informazioni riguardo l'ubicazione della sede
- Contatti telefonici e contatti email

Terminata l'attività di analisi, lo script salva i dati ottenuti sul Database e genera una serie di file in diversi formati che andranno a comporre il Dataset finale.

Capitolo 3

Progettazione e implementazione

Questo capitolo contiene i dettagli relativi alla progettazione e all'implementazione del progetto svolto. Nella prima sezione viene fornita una panoramica sulle tecnologie (linguaggi, applicazioni e librerie) utilizzate, mentre nelle sezioni successive sono forniti i dettagli relativi all'implementazione (codice, algoritmi e utilizzo delle tecnologie descritte).

3.1 Tecnologie utilizzate

L'ambiente di sviluppo che ho utilizzato per la realizzazione di questo progetto è *XAMPP*[1]; si tratta di una piattaforma software che mette a disposizione un *Server Apache HTTP*[2], il *database MySQL*[3] e tutti gli strumenti necessari per utilizzare i linguaggi di programmazione *PHP* e *Perl*.

Tutte le componenti del progetto sono state implementate utilizzando PHP (versione 5.5) come linguaggio di programmazione, fatta eccezione per l'applicazione web, per la quale ho utilizzato *Javascript*.

[1] http://www.apachefriends.org/
[2] http://httpd.apache.org/
[3] http://www.mysql.it/

Per l'installazione di alcune librerie elencate nei prossimi paragrafi, ho utilizzato *Composer*[4], uno strumento per la gestione delle dipendenze in PHP.

3.1.1 Google Custom Search JSON/Atom API

Per la ricerca dei siti relativi alle Associazioni Non Profit ho deciso di appoggiarmi al motore di ricerca di Google. La scelta deriva dal fatto che si tratta di uno dei motori di ricerca più utilizzati e garantisce un'alta precisione nei risultati ottenuti. Inoltre, Google mette a disposizione una serie di API (*Application Programming Interface*), disponibili per un'ampia gamma di linguaggi, che permettono di personalizzare le ricerche ed effettuare le chiamate direttamente all'interno del codice in modo semplice ed efficiente.

Come suggerisce il nome, la Custom Search permette di creare un motore di ricerca personalizzato in base alle proprie esigenze. Tramite l'utilizzo della *Custom Search JSON/Atom API*[5] è possibile sviluppare siti web e applicazioni per recuperare e visualizzare i risultati ottenuti dalle ricerche effettuate per mezzo della *Google Custom Search*.

Configurazione

I prerequisiti necessari per poter utilizzare l'API sono:

- possedere l'ID (identificatore) della Custom Search che si vuole utilizzare;
- possedere un API Key.

Per questo progetto ho creato una Custom Search chiamata *FindOnlus* con le seguenti caratteristiche:

parole chiave Google utilizza queste informazioni per perfezionare i risultati di ricerca e garantire che siano il più pertinenti possibile. Alcune

[4] http://getcomposer.org/

[5] http://developers.google.com/custom-search/

delle parole chiave inserite sono: *'Onlus', 'Non Profit', 'associazioni', 'organizzazioni'*.

versione Per il progetto è stata utilizzata la versione *free* della Custom Search, che permette di effettuare fino a 100 richieste giornaliere.

immagini La ricerca delle immagini è stata disattivata.

siti in cui cercare Non sono stati inseriti siti specifici in cui effettuare le ricerche, che avvengono quindi su tutto il Web.

siti da escludere Anche in questo caso non sono stati specificati particolari siti. Eventuali controlli sui siti da escludere vengono effettuati tramite codice direttamente sui risultati, in base a degli elenchi definiti dal programmatore; questa scelta è stata fatta per consentire, a chiunque voglia utlizzare gli strumenti messi a disposizione, di poter personalizzare ulteriormente le ricerche in base alle proprie esigenze, senza essere limitato da vincoli precedentemente impostati.

Il passo successivo è stato quello di creare un progetto tramite la *Google Developer Console*, in cui è stata inserita la Custom Search appena descritta. A ciascun progetto è associata un'*API Key* che lo identifica univocamente e permette l'accesso alle varie API. Per questo progetto ho creato un API Key di tipo server, configurata con l'indirizzo IP del server su cui risiede l'applicazione.

3.1.2 Google APIs Client library

Questa libreria (scritta in PHP) fornisce l'accesso alla maggior parte delle API di Google. È stata progetta per consentire l'utilizzo, nelle applicazioni server-side, dei servizi forniti dalle API, in modo semplice, flessibile ed efficace.

3.1.3 parseCSV

Alcune funzionalità implementate richiedono l'utilizzo di file CSV (ad esempio l'associazione delle categorie o la ricerca degli indirizzi). Per poter leggere e scrivere questi file ho utilizzato la libreria *parseCSV*[6], che mette a disposizione una serie di funzioni che facilitano e semplificano la gestione di dati in formato CSV.

3.1.4 Simple HTML DOM Parser

Simple HTML DOM[7] è una libreria scritta in PHP che consente la manipolazione dell'HTML; nello specifico, permette di creare degli oggetti di tipo DOM (partendo da un URL o da una stringa) sui quali poi è possibile effettuare delle ricerche.

La ricerca degli elementi all'interno di un oggetto DOM avviene mediante l'uso dei selettori, similmente a jQuery, in una maniera molto semplice e intutiva. Questo è uno dei principali motivi che mi ha spinto al suo utilizzo.

3.1.5 AlchemyAPI

AlchemyAPI[8] mette a disposizione 12 differenti API per l'analisi del testo, che viene effettuata attraverso l'utilizzo di sofisticate tecniche di elaborazione del linguaggio naturale. Ho utilizzato l'API di estrazione delle parole chiave (*Keyword Extraction API*) come strumento per riuscire a determinare una categoria di riferimento per ogni Associazione.

Il funzionamento di base dell'API è molto semplice: analizza un testo (a partire da un URL, un documento HTML o una stringa) e restituisce un insieme di parole chiave, ad ognuna delle quali è associato un punteggio (in termini di rilevanza).

[6] http://github.com/parsecsv
[7] http://simplehtmldom.sourceforge.net/
[8] http://www.alchemyapi.com/

La scelta di utilizzare AlchemyAPI piuttosto che una delle tante altre alternative disponibili, è stata determinata principalmente da 5 fattori:

1. supporto della lingua italiana;
2. la precisione dei risultati forniti;
3. l'ampia varietà delle funzionalità messe a disposizione;
4. la semplicità di utilizzo;
5. la possibilità di utilizzare gratuitamente i servizi offerti.

3.1.6 Stemming

Lo stemming è il processo attraverso il quale si riduce una parola alla sua 'forma radice'. Gli algoritmi di stemming sono molto usati nel campo dell'informatica, ad esempio costituiscono una parte fondamentale dei motori di ricerca che li utilizzano per espandere ed ampliare le interrogazioni. Per questo progetto, la tecnica di stemming è stata utilizzata in combinazione con l'API Keyword Extraction per trovare il tema specifico di ciascuna parola chiave, in modo da poter effettuare dei confronti con dei termini predefiniti considerando solamente la radice della parola ed escludendo tutti i suffissi (ad esempio, grazie allo stemming il confronto tra le parole 'bambini' e 'bambino' sarà positivo).

L'algoritmo di stemming da me utilizzato è stato implementato da *Roberto Mirizzi*[9] ed è specifico per la lingua italiana.

3.1.7 lib-phonenumber

Si tratta di una libreria PHP per l'analisi, la formattazione, la memorizzazione e la convalida di numeri telefonici internazionali. Ho utilizzato

[9]http://www.phpclasses.org/package/3731-PHP-Compute-stem-strings-from-Italian-words.html

lib-phonenmumber[10] per facilitare la ricerca e il riconoscimento dei contatti telefonici di ogni Associazione e in particolare per essere sicura che i risultati trovati corrispondessero a reali numeri di telefono.

3.1.8 EasyRDF PHP

EasyRDF[11] è una libreria progettata per facilitare l'uso e la produzione di grafi RDF. In particolare permette l'analisi e la serializzazione di grafi nei seguenti formati: *RDF, RDF/XML, JSON, Turtle*. Ho utilizzato questa libreria per creare ed esportare il dataset in formato RDF.

3.1.9 MySQL

Per gestire e memorizzare i dati raccolti faccio uso del DBMS (*Database Management System*) MySQL, in quanto offre prestazioni veloci, elevata affidabilità, facilità d'uso ed è la soluzione standard adottata da XAMPP.

Struttura del database

La struttura del database risulta molto semplice e consiste di 6 tabelle:

elenco_siti contiene l'elenco degli URL, a ciascuno dei quali è associato un timestamp che corrisponde alla data dell'ultima analisi effettuata.

categoria contiene l'elenco delle categorie a cui un'associazione può appartenere.

associazione rappresenta la singola associazione e contiene le informazioni ad essa associate.

elenco_email contiene la lista dei contatti email, ognuno associato all'organizzazione a cui appartiene.

[10] https://github.com/giggsey/libphonenumber-for-php
[11] EasyRDF

elenco_numeri contiene la lista dei numeri telefonici, ognuno associato all'organizzazione a cui appartiene.

associazione_categoria rappresenta i collegamenti tra un'associazione e le categorie a cui appartiene.

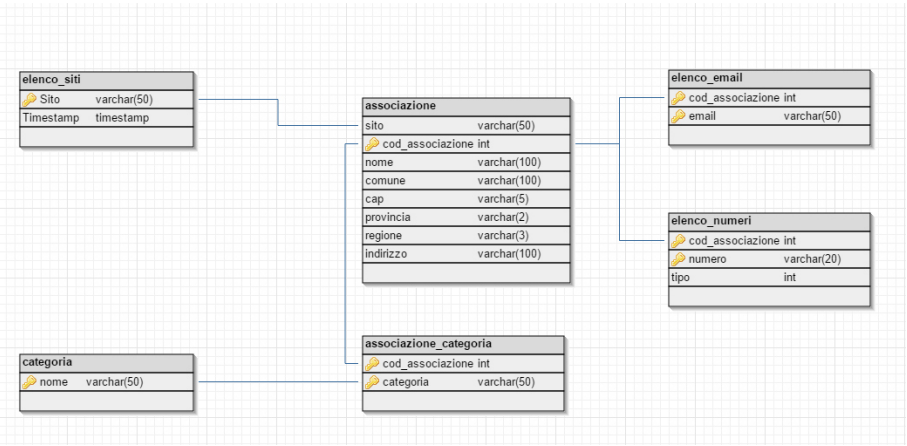

Figura 3.1: Struttura del database

3.1.10 Fuseki Server

Si tratta di un server SPARQL, implementato da *Apache Jena*[12], che permette la gestione di dati in formato RDF.

Grazie a Fuseki è possibile inserire i propri dati in un *triple store*, ovvero un database appositamente studiato per la memorizzazione delle triple RDF; inoltre Fuseki mette a disposizione uno *SPARQL Endpoint* tramite il quale è possibile visualizzare e consultare i dati ed effettuare delle interrogazioni al database tramite query.

Il server può essere lanciato da linea di comando utilizzando lo script *fuseki-server*; le forme più comuni sono:

- `fuseki-server --mem /DatasetPathName` : crea un dataset vuoto;

[12]http://jena.apache.org/

- `fuseki-server --file=FILE /DatasetPathName` : crea un dataset vuoto e successivamente carica FILE al suo interno;
- `fuseki-server --loc=DB /DatasetPathName` : utilizza un TDB esistente, o ne crea uno (vuoto) altrimenti;
- `fuseki-server --config=ConfigFile` : costruisce uno o più servizi in base ad un file di configurazione.

In tutti i casi, */DatasetPathName* rappresenta il nome attraverso il quale sarà possibile accedere al dataset tramite HTTP.

Il server consente operazioni di sola lettura, a meno che non viene specificato come argomento `--update`. È inoltre possibile cambiare la porta su cui gira Fuseki (3030 di default), settando l'argomento `--port=PORT`.

Per accedere al pannello di controllo di Fuseki basta seguire il seguente URL: `http://localhost:3030`; tramite il pannello di controllo è possibile selezionare il dataset e successivamente eseguire una serie di operazioni:

SPARQL Query permette di interrogare il dataset, tramite l'utilizzo di query (endpoint: `http://localhost:3030/ds/query`).

SPARQL Update permette di modificare, aggiornare o eliminare i dati, sempre tramite l'utilizzo di query (endpoint: `http://localhost:3030/ds/update`).

File upload consente di caricare dei file contenenti delle triple RDF.

3.2 Primi passi verso l'implementazione

3.2.1 Analisi della struttura dei siti Web

Prima di passare alla realizzazione e all'implementazione del progetto ho dovuto svolgere un lavoro di analisi relativamente ai siti Web, per riuscire a capire come poter estrapolare le informazioni ricercate. Sono partita dall'elenco disponibile al seguente indirizzo *http://www.nonprofit.viainternet.org/*

e ho recuperato un certo numero di 'siti campione'. L'analisi strutturale effettuata su questi siti, ha fatto emergere numerose analogie riguardo la pubblicazione delle informazioni che mi interessava recuperare:

informazioni base solitamente è presente una pagina *'chi siamo'* in cui sono riportate le informazioni sull'associazione (ad esempio di cosa si occupano, gli interessi).

contatti i numeri telefonici e gli indirizzi email sono, nella maggior parte dei casi, riportati nella pagina *'contatti'*.

indirizzo e sede le informazioni riguardo l'ubicazione sono solitamente riportate nelle pagine *'dove siamo'* o *'dove trovarci'*.

Tutte queste pagine sono raggiungibili tramite la selezione di specifiche voci, del menù presente nell'homepage. In alcuni casi inoltre, le informazioni relative ai contatti o al luogo sono fornite direttamente nel footer della home.

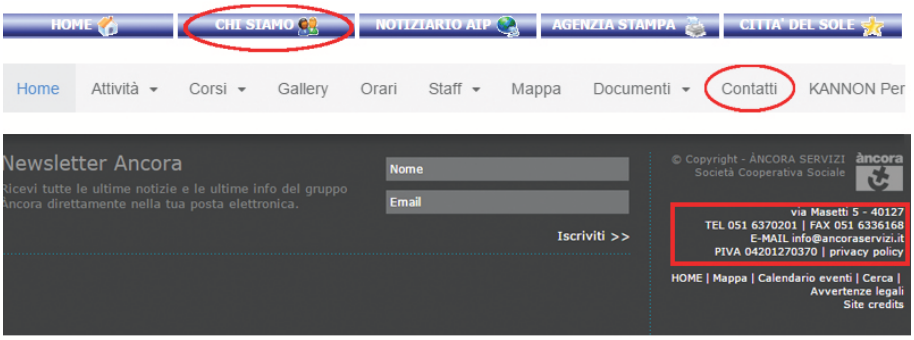

Figura 3.2: Alcuni esempi di pubblicazione delle informazioni

Le informazioni raccolte durante questo lavoro di analisi sono state necessarie per capire come implementare l'algoritmo di scraping nel modo più efficiente possibile.

3.2.2 Gestione delle categorie

Ho stilato una lista di categorie, prendendo come punto di riferimento gli elenchi che si trovano online; le categorie considerate sono riportate in tabella 3.1 .

Arte e cultura	Associazione di donatori
Commercio equo-solidale	Disabili e Handicap
Donne	Famiglia
Istruzione e sostegno	Lavoro
Medicina	Musica
Protezione civile	Recupero e assistenza
Ricerca scientifica	Spettacolo
Sport	Tutela degli animali
Tutela dei cittadini	Tutela dei minori e adozioni
Tutela dell'ambiente	Volontariato

Tabella 3.1: Elenco delle categorie utilizzate

A ciascuna categoria sono state associate una serie di parole chiave, che saranno utilizzate in fase di analisi dei siti Web per riuscire a determinare a quale categoria fanno riferimento. L'elenco delle categorie è stato inserito sul database nella tabella *categorie*, mentre le parole chiave sono contenute in un file CSV, la cui struttura consiste di tre campi: *codice,categoria,keyword*. Ho scelto di inserire queste informazioni su un file CSV in modo che possano essere modificate facilmente in futuro.

3.3 Implementazione del Crawler: ricerca dei siti

Il Crawler implementato, è di tipo *focused* e permette quindi di effettuare delle ricerche mirate, su tutto il Web, in base a gli input inseriti. I fattori

ritenuti fondamentali nella progettazione riguardavano la possibilità di esecuzione in background e da linea di comando, cercare di rendere il processo il più automatizzato possibile, consentire di differenziare la ricerca in base alla regione o alla provincia e cercare di ottenere il maggior numero di risultati attinenti al contesto.

I file principali realizzati per l'implementazione del Crawler sono:

ricerca_siti.php script da eseguire per iniziare la ricerca dei siti.

esamina_siti.php file che contiene le funzioni relative all'analisi dei siti Web

In figura 3.3 è riportato uno schema relativo alla struttura e al funzionamento del Crawler.

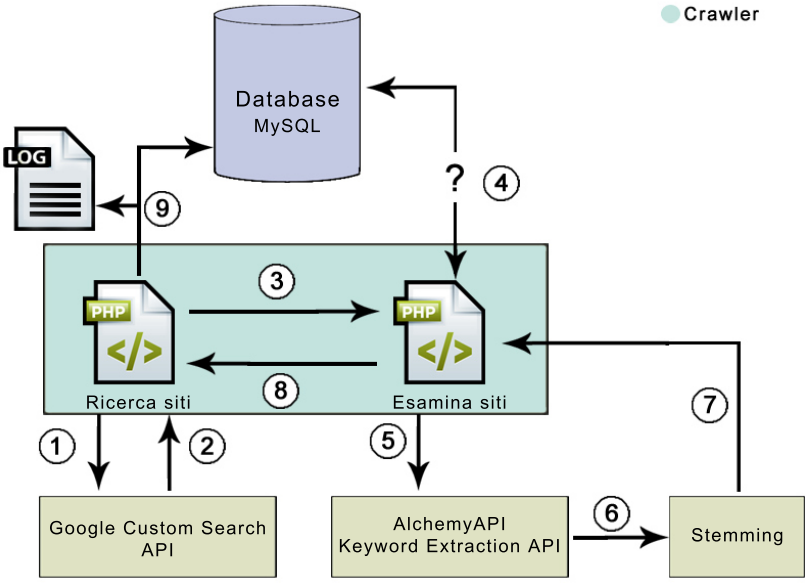

Figura 3.3: Struttura e funzionamento del Crawler

Lo script viene lanciato inserendo una serie di argomenti, che rappresentano i termini della ricerca che si vuole effettuare. All'interno del codice viene subito effettuato un controllo, per determinare se lo script è stato lanciato

da linea di comando o da browser, in modo da sapere come gestire gli input inseriti.

Se lo script è stato lanciato in maniera corretta, viene richiamato il servizio *Custom Search*, sfruttando le classi messe a disposizione dalla libreria *Google APIs Client*.

```
1 $client = new Google_Client();
2 $client->setApplicationName("findOnlus");
3 $client->setDeveloperKey(API_KEY);
4 $search = new Google_Service_Customsearch($client);
```

`Google_Client` contiene i sistemi di IO, autenticazione e altre classi necessarie per il funzionamento dei servizi; un'istanza del client è richiesta come parametro per l'istanziazione dell'oggetto `Google_Service_Customsearch`.

La ricerca viene eseguita chiamando il metodo `listCse`. Ogni chiamata effettuata all'API Custom Search permette di ottenere un massimo di 10 risultati, per questo motivo la chiamata è stata inserita all'interno di un ciclo `for` ripetuto 4 volte, dove ad ogni iterazione viene incrementata la variabile `$start`; questo limite deriva da una scelta basata su dei test effettuati, in cui ho notato che dopo i primi 40 risultati cominciava ad esserci sempre meno attinenza rispetto ai termini ricercati.

```
1 for($i=0;$i<4;$i++){
2         $start = ($i*10)+1;
3         $result = $search->cse->listCse($filter, array(
4         'cx' => "CX_CUSTOM_SEARCH",'start'=>$start
5         ));
6         ...
7 }
```

Alla funzione sono passati i seguenti parametri: la stringa su cui effettuare la ricerca (`$filter`), un array contenente l'identificatore della Custom Search descritta nella sezione precedente (`cx`) e un intero che rappresenta l'offset iniziale da cui devono partire i risultati restituiti (`start`).

La funzione restituisce un oggetto di tipo JSON (`$result`) contenente una serie di dati; la parte che ci interessa è strutturata nel modo seguente:

```
...
"items": [
{
 "kind": "...",
 "title": "...",
 "htmlTitle": "...",
 "link": "...",
 "displayLink": "...",
 "snippet": "...",
 "cacheId": "...",
 "mime": "...",
 "fileFormat": "...",
 "formattedUrl": "...",
 "htmlFormattedUrl": "...",
 "pagemap": {...}
},
...
]
...
```

Partendo da `$result` viene popolato un array (`$elenco_siti`) con l'URL associato a ciascun item (contenuto del campo `formattedUrl`).

A questo punto, occorre effettuare un'analisi su ciascun URL, per decidere se inserirlo nell'elenco oppure scartarlo. Un URL deve essere scartato nei seguenti casi:

- il sito Web non risulta raggiungibile;
- l'URL è già presente in elenco;
- il sito Web non può essere utilizzato ai fini di questo progetto o non riguarda un'associazione non profit.

Per questo motivo ho implementato la funzione `esamina($elenco_siti)`, che prende come parametro l'array contenente gli URL dei siti Web e per ognuno di essi effettua una serie di operazioni. Per prima cosa risalgo all'homepage del sito eliminando dall'URL eventuali path in eccesso: si tratta di un'operazione necessaria in quanto non sempre i risultati restituiti dalla ricerca tramite Custom Search rimandano alla pagina principale. Successivamente viene effettuato un controllo per verificare che l'URL non sia già presente sul database. Questo controllo viene eseguito chiamando la funzione `cerca_sito($link)` che si collega al database per verificare l'eventuale presenza dell'URL passatogli come parametro.

```
1 function cerca_sito($link){
2         $db = new Db();
3         $res = $db->select("SELECT Sito from elenco_siti
              where sito='".$link."';");
4         if(count($res)>1) return true;
5         return false;
6 }
```

`Db` è una classe creata appositamente per gestire la connessione con il database e contiene i metodi `query($q)` e `select($q)` con i quali è possibile rispettivamente modificare e interrogare il database.

Prima di procedere con una breve analisi del sito, viene effettuato un ulteriore controllo sull'URL per verificare che non appartenga alla cossidetta *black list*. La 'lista nera' altro non è che un file CSV contenente un elenco di domini che non si vogliono includere nella ricerca. Il controllo viene effettuato aprendo in lettura il file `black_list.csv` (sfruttando la libreria *parseCSV*) e iterando sull'elenco alla ricerca di un matching con l'URL che si sta analizzando: se si ottiene un riscontro positivo, l'URL viene scartato.

La fase successiva consiste nell'analisi del sito. Nello specifico, lo scopo di questo passaggio è quello di riuscire a determinare, tramite un'analisi del contenuto del sito Web, se si tratta di un sito di un'associazione non profit

oppure no. L'idea generale consiste nell'utilizzare l'API di estrazione delle parole chiave, per effettuare un'analisi sul testo e cercare una corrispondenza con delle keywords contenute in un array (`$parole`) da me definito.

```
1  $html = file_get_html($link);
2  if(is_object($html)){
3    $alchemyapi = new AlchemyAPI();
4    $link_descrizione = $html->find("a[href*=siamo],a[href
       *=storia],a[href*=associazione]");
5      if(count($link_descrizione) > 0){
6        ...
7        $url = $link_descrizione[$i];
8      }
9      else{
10       $link_descrizione = $html->find("a");
11       foreach($link_descrizione as $a){
12         if(stripos(strtolower($a->innertext),"chi siamo"
             ) !== false || stripos(strtolower($a->
             innertext),"storia") !== false ...){
13           $url = $a->href;
14           break;
15         }
16       }
17     }
18   ...
19   if($url == ""){
20     $response = $alchemyapi->keywords('url', $link,
         array('maxRetrieve'=>20));
21   }
22   else{
23     $url = get_absolute_url($url,$link);
24     $response = $alchemyapi->keywords('url', $url, array
         ('maxRetrieve'=>20));
25   }
```

Per prima cosa, provo a recuperare il contenuto della pagina Web chiamando la funzione `file_get_html` (della libreria *Simple HTML DOM*); la funzione effettua una chiamata HTTP all'URL passatogli come parametro e restituisce un oggetto di tipo DOM che rappresenta la pagina HTML o `NULL` nel caso non riesca ad instaurare la connessione. Tramite la funzione `find` viene poi effettuata una ricerca degli `anchor node` il cui attributo `href` contiene almeno una parola tra *'siamo','storia','associazione'*. Se non ottengo risultati, effettuo una ricerca più generica su tutti gli `anchor node`, effettuando poi un controllo sul contenuto dell'`innertext`. In questo modo, cerco di risalire alla pagina in cui è probabile che sia riportata una descrizione del sito. A questo punto utilizzo un'istanza di *AlchemyAPI* (`$alchemyapi`) per analizzare il contenuto della pagina trovata ed estrarre le parole chiave; alla funzione `keywords` vengono passati i seguenti parametri: una stringa che indica la fonte da cui recuperare il testo ('url'), l'URL della pagina da cui voglio etrapolare le informazioni, il numero di risultati che voglio ottenere (20). La risposta consiste in un array contenente le parole trovate, ordinate per rilevanza. Infine, le keywords restituite da AlchemyAPI vengono ricercate all'interno dell'array `$parole` per mezzo della funzione `cerca_corrispondenza($keys_trovate,$keys_fornite)`; prima di effettuare i confronti, alle parole viene applicata la tecnica di *stemming*, in modo da risalire alla loro radice per effettuare dei controlli più accurati. La ricerca infatti non deve essere influenzata dalla forma in cui una determinata parola è espressa (ad esempio plurale piuttosto che singolare), ma dal significato che essa assume. Lo stemming viene eseguito attraverso l'utilizzo di un'istanza della classe `ItalianStemmer`. Se l'array restituito dalla funzione `keywords` è vuoto o non vengono trovate corrispondenze, l'URL è scartato.

Completata l'analisi, i siti ritenuti 'corretti' vengono salvati e inseriti sul database nella tabella *elenco_siti*, associandogli un valore di `timestamp = NULL`, per indicare che il sito non è ancora stato analizzato in maniera approfondita (per il recupero delle informazioni).

Prima di terminare la sua esecuzione, lo script genera un file di log in cui sono riportati i dati ottenuti, ovvero l'elenco degli URL salvati e quelli scartati, con i relativi totali. Il file è generato semplicemente utilizzando le funzioni `fopen` (per creare ed aprire il file in scrittura) e `fwrite` (per scrivere le righe nel file di testo).

3.4 Implementazione dello Scraper: ricerca delle informazioni

Lo Scraper è il componente che si occupa di recuperare le informazioni ricercate dai siti trovati grazie al processo di crawling.

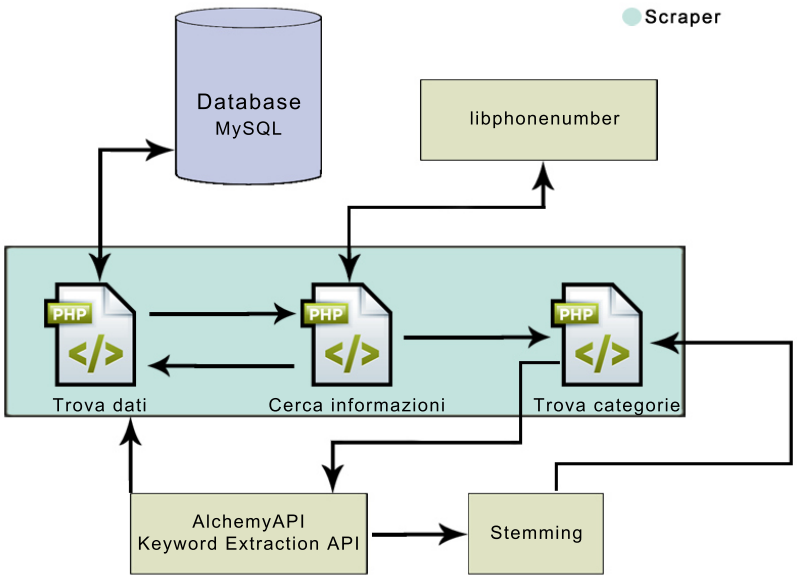

Figura 3.4: Struttura e funzionamento dello Scraper

Per prima cosa viene recuperato l'elenco degli URL dal database eseguendo la seguente query:

```
SELECT * from elenco_siti WHERE Timestamp is null or
   TIMESTAMPDIFF(MONTH,Timestamp,now())>0;
```

In questo modo vengono considerati solo gli URL relativi a siti che non sono mai stati analizzati o per cui è trascorso più di un mese dall'ultima analisi; questo per evitare che ad ogni esecuzione venga scansionata l'intera lista.

Per ogni URL restituito viene chiamata la funzione `findInformation`, tramite la quale viene effettuata la ricerca delle informazioni. I dati raccolti per ogni associazione, vengono salvati in un array così strutturato:

```
/*Array che contiene tutti i dati*/
$associazione['link'] = "URL DEL SITO WEB";
$associazione['nome'] = "NOME ASSOCIAZIONE";
$associazione['email'] = $email;
$associazione['numeri'] = $numeri;
$associazione['luogo'] = $luogo;
$associazione['categorie'] = $categorie;
/*Array contenente gli indirizzi email*/
$email[0] = "email 1";
$email[n] = "email n";
/*Array contenente i numeri telefonici*/
$numeri[0] = "numero 1";
$numeri[n] = "numero n";
/*Array contenente le categorie*/
$categorie[0] = "categoria 1";
$categorie[n] = "categoria n";
/*Array contenente le informazioni sul luogo*/
$luogo['comune'] = "COMUNE";
$luogo['cap'] = "CAP";
$luogo['provincia'] = "PROVINCIA (SIGLA)";
$luogo['regione'] = "REGIONE (SIGLA)";
$luogo['indirizzo'] = "INDIRIZZO E NUMERO CIVICO";
```

I passaggi effettuati all'interno della funzione sono, nell'ordine:

1. Ricerca delle categorie;
2. Ricerca del nome dell'associazione;
3. Ricerca delle informazioni di contatto (email, numeri di telefono, indirizzo).

La ricerca delle categorie avviene per mezzo della funzione `trova_categorie($link)` (definita all'interno del file *trova_categorie.php*).

```
1  $stemmer = new ItalianStemmer();
2  ...
3  function trova_categorie($link){
4    global $stemmer;
5    $categorie = get_category_list();
6    $keywords = get_keywords($link);
7    if($keywords != null && array_key_exists('keywords',
       $keywords) === true){
8      $result = associa_categoria($keywords,$categorie);
9      if(count($result)>0)
10       return $result;
11     else
12       return null;
13   }
14   return null;
15 }
```

La variabile `$categorie` contiene l'elenco delle categorie e delle relative parole chiave, che sono definite all'interno del file *categorie.csv*; questi dati vengono recuperati attraverso la funzione `get_category_list()` il cui codice è riportato di seguito:

```
1  function get_category_list(){
2    $categorie = array();
3    $last_codice = "";
4    $csv_file = new parseCSV();
```

```
5   $csv_file->auto('../src/categorie.csv');
6   foreach ($csv_file->data as $key => $row){
7     $codice = $row['categoria'];
8     if(strcmp($codice,$last_codice) == 0)
9       array_push($categorie[$codice],$row['keyword']);
10    else{
11      $categorie[$codice] = array();
12      array_push($categorie[$codice],$row['keyword']);
13      $last_codice = $codice;
14    }
15  }
16  return $categorie;
17 }
```

Quello che si ottiene è un array associativo che risulta così strutturato:

```
$categorie['categoria 1'] = {'key_1',...,'key_n'};
...
$categorie['categoria n'] = {'key_1',...,'key_n'};
```

La variabile `$keywords` contiene l'elenco delle parole chiave ottenute per mezzo dell'API AlchemyAPI, chiamata dalla funzione `get_keywords`. La ricerca avviene in maniera analoga a quella effettuata dall'algoritmo di crawling e descritta nella sezione precedente (3.3). Ottenuti i due insiemi di parole chiave, viene chiamata la funzione `associa_categoria($keywords, $categorie)` che cerca delle corrispondenze. Nello specifico, viene effettuato un confronto (sfruttando la tecnica di stemming) tra le parole chiave di ciascuna categoria e l'elenco di parole chiave restituito da Alchemyapi; ad ogni categoria è associato un contatore che viene incrementato ogni volta che viene trovata una corrispondenza. La funzione restituisce un array associativo `($result)` che contiene l'elenco delle categorie e il punteggio a loro associato.

L'associazione a questo punto è associata alle categorie (una o più) che hanno ottenuto il punteggio massimo. Per evitare casi in cui un'associazione

risulti non appartenere a nessuna categoria, è stata creata una categoria di default che è *'Associazione Non Profit'*: tutte le associazioni considerate appartengono a questa categoria.

Il passo successivo consiste nel determinare il nome dell'associazione. Questo avviene effettuando una ricerca sui tag `title`, all'interno del DOM. Dai risultati ottenuti viene estrapolato l'`innertext` che successivamente viene formattato in modo da eliminare eventuali caratteri speciali, spazi multipli o parti del testo che non ci interessano; la formattazione della stringa avviene mediante l'utilizzo della funzione `preg_replace(REGEX, PLACE_HOLDER, STRINGA)`.

```
1  $html = file_get_html($link);
2  ...
3  $associazione['nome'] = parse_url($link,PHP_URL_HOST);
4  foreach($html->find("title") as $element){
5    $titolo = $element->plaintext;
6    if($titolo != "" && strcasecmp($titolo,"home") != 0){
7      /*Elimino eventuali caratteri speciali*/
8      $titolo = preg_replace('/s*(home|homepage|home page|
           index)\s*/',"",$titolo);
9      $titolo = preg_replace('/\s{2,}/',' ',$titolo);
10     ...
11   }
12 }
```

Se la ricerca restituisce un insieme vuoto allora viene utilizzato l'host dell'URL come nome per l'associazione.

L'ultimo punto consiste nella ricerca delle informazioni di contatto e indirizzo, il cui procedimento è descritto nella sezione 3.4.1.

Terminata la fase di analisi i dati ottenuti vengono salvati, distinguendo due possibili casi: se si tratta della prima analisi effettuata sul sito web, i

dati vengono semplicemente inseriti sul database per mezzo della funzione `inserisci_dati($associazione)`, a cui è passato come parametro l'array contenente tutte le informazioni raccolte; se invece il sito era già stato precedentemente analizzato, viene effettuato un confronto tra i dati già presenti sul database e i dati appena raccolti. La funzione che si occupa di effettuare l'aggiornamento dei dati è la seguente:

```
1  function aggiorna_dati($link,$info_nuove,$info_vecchie){
2    ...
3    /*Verifica delle informazioni sull'email*/
4    if(array_key_exists("email",$info_nuove)){
5      $email = $info_nuove['email'];
6      if(array_key_exists("email",$info_vecchie)){
7        foreach($info_vecchie['email'] as $e){
8          if(array_search($email,$e) === false)
9            $db->query("DELETE FROM elenco_email where
                 email='".$e."';");
10       }
11     }
12     foreach($email as $e){
13       if(array_search($info_vecchie['email'],$e) ===
            false)
14         $db->query("INSERT INTO elenco_email VALUE('".
             $link."', '".$e."');");
15     }
16   }
17   ...
18 }
```

`$info_nuove` è l'array contenente le informazioni trovate, `$info_vecchie` è l'array contenente i dati già presenti sul database, relativamente ad una precisa associazione. Quello che viene effettuato è un controllo dell'email presenti sul database, che vengono ricercate tra i contatti recuperati con l'ultima analisi effettuata; se la ricerca è negativa, l'email viene cancellata dal

database. Infine, vengono inserite tutte le nuove email trovate. Il procedimento descritto è analogo per quanto riguarda l'aggiornamento di tutte le altre informazioni.

3.4.1 Recupero dei contatti e dell'indirizzo

Le informazioni relative ai contatti e al luogo sono solitamente riportate nelle pagine denominate 'contatti', 'dove trovarci, 'dove siamo' o simili. Per questo motivo, il primo passo effettuato consiste nel ricercare nel DOM tutti gli `anchor node` il cui attributo `href` contenga determinate stringhe:

```
1  $pag_contatti = $html->find("a[href*=contatt] , a[href*=
       contact], a[href*=dove], a[href*=siamo], a[href*=sede
       ]");
2  ...
3  foreach($pag_contatti as $element){
4    $link_contatti = $element->href;
5    /*Trasformo i link relativi in assoluti*/
6    $link_contatti = get_absolute_url($link_contatti,
         $dominio);
7    ...
8    $associazione = findContactInformation($link_contatti,
         $associazione);
9    ...
10 }
```

Un aspetto fondamentale di cui bisogna tener conto è la possibilità che negli attributi href siano riportati degli URL relativi; in questo caso occorre trasformare i link relativi in assoluti ed è questo il lavoro svolto dalla funzione `get_absolute_url` che prende come parametri l'URL che si vuole trasformare e il dominio del sito web a cui appartiene.

Successivamente viene effettuata una chiamata alla funzione `findContactInformation` che si occupa di ricercare le informazioni di contatto all'interno della pagina che fa riferimento all'URL passato come pa-

rametro (`$link_contatti`). Tramite la funzione `file_get_html` viene recuperato il contenuto della pagina, restituito sottoforma di un oggetto DOM. Dal DOM viene estrapolato l'innertext, sul quale si andanno poi ad eseguire una serie di ricerche sfruttando l'uso delle regex: l'email, i numeri di telefono e gli indirizzi hanno delle strutture precise e ben definite che possono essere identificate facilmente attraverso la definizione di particolari espressioni regolari.

Ricerca delle email

Un indirizzo email è così composto: *nomeUtente@dominio*. Il nomeUtente può contenere qualsiasi carattere alfabetico e numerico (vocali accentate escluse) e alcuni simboli, come ad esempio l'underscore (_) o il punto.

L'espressione regolare utilizzare per il matching delle email è la seguente: `([\+\.]*@[\w+\.]*[\w+\-\w+]*\.\w{2,3})`.

Oltre a ricercare gli indirizzi email espressi in forma testuale, viene effettuata una ricerca anche sugli `anchor node` con un particolare tipo di collegamento ipertestuale: `mailto`. Molte associazioni infatti non scrivono esplicitamente i propri indirizzi, ma utilizzano link di questo tipo per consentire ai visitatori di contattarli, semplicemente con un 'click'. Ovviamente, per evitare risultati doppi viene effettuato un controllo prima di inserire ciascuna email all'interno dell'array.

Il meccanismo appena descritto è implementato per mezzo del seguente codice:

```
1 preg_match_all("/([\w+\.]*@[\w+\.]*[\w+\-\w+]*\.\w{2,3})
     /is",$content,$addresses);
2 if(count($addresses[0]) > 0){
3   $associazione['email'] = array();
4   foreach($addresses[1] as $curEmail) {
5     $curEmail = preg_replace('/\s{2,}/',' ',$curEmail);
6     if(array_search (trim($curEmail," "),$associazione['
        email']) === false)
```

```
7         array_push($associazione['email'],trim($curEmail,"
              "));
8     }
9   }
10  ...
11  foreach($html->find("a[href^=mailto:]") as $element){
12    $result = array();
13    $email = str_replace("\%20","",$element->href);
14    preg_match("/([\w+\.]*@[\w+\.]*[\w+\-\w+]*\.\w{2,3})/
          is",$email,$result);
15    $email = $result[0];
16    if(!empty($email)){
17      if(array_key_exists("email",$associazione) === false
            )
18        $sito['email'] = array();
19      if(array_search ($email,$associazione['email']) ===
            false)
20        array_push($associazione['email'],$email);
21    }
22  }
```

Ricerca dei numeri telefonici

Rispetto agli indirizzi email, scrivere un'espressione regolare per la descrizione dei numeri telefonici è risultato più complesso. La difficoltà maggiore è legata agli svariati modi in cui un numero può essere scritto:

333-1234567	3331234567
333.1234567	333 1234567
333 12 34 567	+39 3331234567
039 333 1234567	etc.

Tabella 3.2: Esempi di scrittura numeri telefonici

I numeri riportati sopra sono scritti in maniera differente, ma rappresentano tutti lo stesso numero telefonico. Inoltre bisogna considerare anche la differenza tra numeri fissi e mobili e la possibilità che essi includano dei prefissi internazionali o meno.

Nello scrivere l'espressione regolare utilizzata ho cercato di includere il maggior numero di casi possibili, effettuando molti test e aiutandomi con delle ricerche online sull'argomento. Il risultato ottenuto è stato il seguente: `\(?\s?\d{3,4}\s?[\)\.\-]?\s*\d3\s*[\-\.]?\s*\d{3,4}`.

In questo caso, l'utilizzo delle sole espressioni regolari come mezzo per la ricerca dei numeri telefonici non è risultato sufficiente. Infatti, tramite l'espressione riportata venivano recuperati anche altri tipi di numeri che rientrano nella struttura descritta, come ad esempio la partita IVA.

Questo problema l'ho risolto implementando un algoritmo in grado di riconoscere i numeri di partita IVA; questi numeri infatti sono creati seguendo una precisa logica e quindi, effettuando una serie di operazioni e controlli, è possibile riuscire a determinare se un numero di 11 cifre corrisponde ad una partita IVA oppure no. È possibile trovare una descrizione formale dell'algoritmo su *Wikipedia* [13], mentre la mia implementazione è stata la seguente:

```
1  function is_partita_iva($numero){
2    if(strlen($numero) == 11 && strrpos($numero," ") ===
        false){
3      $cifre_dispari= 0;
4      $cifre_pari = 0;
5      $array_num = str_split($numero);
6      $car_controllo = intval($array_num[count($array_num)
          -1]);
7      for($i=0;$i<9;$i=($i+2)){
8        $cifre_dispari = $cifre_dispari+intval($array_num[
            $i]);
9      }
10     for($i=1;$i<11;$i=($i+2)){
```

```
11        $mol = intval($array_num[$i])*2;
12        if($mol >9)
13          $mol = $mol-9;
14        $cifre_pari = $cifre_pari+$mol;
15      }
16      $result = ($cifre_dispari+$cifre_pari)\%10;
17      if($car_controllo == 0 && $result == 0)
18        return true;
19      else if ($car_controllo != 0){
20        $result = 10-$result;
21        if($car_controllo == $result)
22          return true;
23      }
24      return false;
25    }
26    return false;
27  }
```

Per riuscire a determinare ulteriori informazioni riguardo ai numeri telefonici ho utilizzato la libreria *libphonenumber*. Attraverso un'istanza di `PhoneNumberUtil` è possibile analizzare un numero ed ottenere una serie di informazioni riguardo:

- la validità del numero;
- la tipologia (mobile, fisso, numero verde);
- lo stato a cui è associato.

Ricerca dell'indirizzo

Riuscire a gestire tutti i casi legati alla scrittura di un indirizzo risulta molto difficile, se non quasi impossibile, in quanto occorre tener conto di moltissimi fattori. Ho effettuato molti tentativi ed eseguito numerose prove

per cercare di definire un'espressione regolare che mi permettesse di racchiudere il maggior numero di casi possibile. Alla fine ho deciso di prendere come punto di riferimento lo standard di composizione degli indirizzi indicato dalle *Poste Italiane*[17], secondo cui un indirizzo deve contenere nell'ordine:

- nome della via;
- numero civico;
- cap;
- comune;
- provincia (sigla).

In particolare, il CAP costituisce il punto fondamentale attraverso il quale poter identificare un indirizzo, in quanto è l'unico elemento che ha una struttura precisa e su cui è possibile effettuare delle verifiche riguardo la validità.

Il codice relativo alla ricerca di un indirizzo è il seguente:

```
1  ...
2  preg_match_all('/(via|corso|piazza|viale)[a-zA-Z0-9\s
       ,.-]*\d{2}[01589]\d{2},{0,1}[a-zA-Z0-9\s,.-]*((\([A-Z
       ]{2}\))|(bologna))/i',$content,$indirizzi);
3  if(count($indirizzi[0]) > 0){
4    $associazione['luogo'] = array();
5    foreach($indirizzi[0] as $ind) {
6      $form_ind = formatta_indirizzo($ind);
7      $associazione['luogo'] = $form_ind;
8    }
9  }
10 ...
```

La funzione `formatta_indirizzo` scompone l'indirizzo trovato nei vari campi di cui è composto e restituisce un'array associativo popolato con i

dati ottenuti. Ad esempio, partendo dall'indirizzo *Via esempio 1, 40100 Comune (AA)* quello che si ottiene è un array così composto:

```
$array['comune'] = "Comune"
$array['provincia'] = "AA"
$array['cap'] = "40100"
$array['indirizzo'] = "Via esempio 1"
```

Nel caso non venisse trovato alcun indirizzo, viene effettuata una ricerca solamente sul CAP, in modo da avere almeno un riferimento riguardo la sede dell'associazione. Attraverso il CAP infatti si riesce facilmente a risalire al Comune, la Provincia e la Regione associate. Per fare questo utilizzo un file CSV contenente l'elenco dei comuni italiani e relative informazioni (tra le quali CAP,Provincia e Regione). Il file, denominato `listacomuni.csv` è messo a disposizione dall'organizzazione *LAB di Comuni-Italiani* [14].

```
1  ...
2  preg_match_all('/\s\d{2}[01589]\d{2},{0,1}\s/i',$content
       ,$indirizzi);
3  if(count($indirizzi) > 0){
4    $file= new parseCSV();
5    $file->parse('../src/listacomuni.csv');
6    $associazione['luogo'] = array();
7    foreach($indirizzi as $ind) {
8      foreach ($c->data as $key => $row){
9        $cap = $row['CAP'];
10       if (strpos($cap,'x') !== false){
11         $index = strpos($cap,'x');
12         $cap_pre = substr($cap,0,$index);
13         $ind_pre = substr($ind,0,$index+1);
14         if(intval($cap_pre) == intval($ind_pre)){
15           /*Aggiungo i dati*/
16         }
17       }
```

```
18        else if(intval($cap) == intval($ind)){
19          /*Aggiungo i dati all'array $associazione*/
20        }
21          ...
```

I CAP trovati tramite l'espressione regolare definita e l'utilizzo della funzione `preg_match_all` vengono ricercati all'interno dell'elenco contenuto nel file CSV. Alcuni Comuni, come ad esempio Bologna, hanno CAP diversi a seconda del quartiere o della zona a cui fanno riferimento; in questi casi nel file è indicato un CAP generico, in cui le cifre variabili sono specificate tramite il carattere 'x' (esempio: 401xx). Per questo motivo viene effettuato il controllo della riga numero 10: se il CAP contiene una 'x', il confronto viene effettuato solo sulle prime cifre del Codice di Avviamento Postale.

3.4.2 Produzione del dataset

La creazione del dataset avviene al termine dell'esecuzione dello script di Scraping. Il dataset è prodotto nei seguenti formati:

- CSV
- JSON
- RDF (Turtle)

Dataset CSV

Il dataset in formato CSV viene creato dalla funzione `crea_data_csv` ed è composto da una serie di file: *associazioni.csv, elenco_email.csv, elenco_numeri.csv, associazioni_categorie.csv, elenco_siti.csv.* In pratica, ogni file corrisponde ad una tabella del database: tramite delle semplici query viene recuperato il contenuto di ogni tabella, che viene poi inserito riga per riga all'interno del file di riferimento.

Dataset JSON

Il dataset in formato JSON viene creato dalla funzione `crea_data_json` che recupera dal database l'elenco delle associazioni e delle relative informazioni e crea il file *associazioni.json* così composto:

```
[{"nome":"...",
  "link":"...",
  "luogo":{
    "comune":"...",
    "cap":"...",
    "provincia":"...",
    "regione":"..."
  },
  "categoria":[
    "...",
    "..."
  ]},
  ...
]
```

Dataset RDF

Il dataset in formato RDF viene creato dalla funzione `crea_data_rdf` sfruttando il file JSON precedentemente creato.

Nel seguente codice sono riportati solamente i punti principali, per dare un'idea di come avviene la creazione del grafo mediante l'utilizzo della libreria *EasyRDF*.

```
1 $graph = new EasyRdf_Graph();
2 EasyRdf_Namespace::set('org', 'http://www.w3.org/ns/org#
    ');
3 ...
4 foreach ($elenco_associazioni as $element){
5   ...
```

```
6    $associazione = $graph->resource($iri_associazione, '
        org:Organization');
7    $associazione->set("skos:prefLabel",$nome_associazione
        );
8    $associazione->set("foaf:homepage",$url_associazione);
9    $associazione->addResource("org:purpose",
        $uri_categoria));
10   ...
11   /*Creazione delle risorse in base ai dati dell'
        associazione*/
12 }
13 $string = $graph->serialise("turtle");
14 $dir_path = '../data/rdf';
15 $file = fopen($dir_path."/data_rdf.ttl", "w");
16 fwrite($file,utf8_encode($string));
17 fclose($file);
```

Capitolo 4

Un caso di studio: No-Profit in provincia di Bologna e in Emilia Romagna

Per questo lavoro di tesi ho preso in considerazione il mondo del Non Profit a livello della provincia di Bologna, in quanto si tratta di una realtà sulla quale potevo avere un maggior controllo e che mi permetteva di poter effettuare in maniera molto semplice delle verifiche sui risultati ottenuti.

Il processo che mi ha portato alla realizzazione del dataset risulta suddiviso in varie fasi. Il primo passo effettuato è stato quello di cercare i siti web delle associazioni; questo è avvenuto tramite l'esecuzione del Crawler: lo script è stato lanciato ed eseguito più volte, sia da terminale che browser, inserendo input di ricerca diversi e specifici per il contesto considerato. Ogni esecuzione restituiva 40 URL, dei quali circa 15 venivano scartati, principalmente per i seguenti motivi:

- Non inerenti con la ricerca effettuata;
- Lo script non riusciva a prelevare i dati dalle pagine;
- Il sito era già presente in elenco.

Questa fase ha richiesto comunque un controllo manuale finale, per verificare che i Siti Web ritenuti 'buoni' fossero effettivamente utili allo scopo.

Successivamente ho lanciato lo Scraper per recuperare le informazioni contenute nei Siti Web presenti nell'elenco ottenuto. Lo script è stato lanciato più volte e in alcuni casi anche tramite l'esecuzione di più istanze in parallelo, per cercare di velocizzare il processo di estrazione dei dati; durante i vari test effettuati, lo script è stato in grado di analizzare, in media, circa 10 siti al minuto. Durante questa fase lo script è stato configurato in modo da scartare automaticamente tutti i Siti Web di associazioni che non risultavano avere sede a Bologna e provincia, in quanto non interessanti per il caso considerato. Anche in questo caso sono stati necessari dei controlli manuali, per verificare la correttezza dei dati ottenuti e nel caso effettuare degli aggiustamenti.

Il passo finale è stato quello di formattare i dati raccolti, creando le triple RDF che sono state infine caricate sul triple store.

Per aumentare la dimensione del dataset, ho cercato di integrare i dati da me raccolti tramite i tool sopra descritti, con dati provenienti da altre fonti. Questo è avvenuto tramite l'implementazione di script creati ad-hoc per cercare di estrapolare (tramite scraping) le informazioni contenute in determinati Siti Web contenenti elenchi di associazioni.

Nelle sezioni successive sono riportate alcune considerazioni riguardo i dati ottenuti. In particolare viene prima fornita una valutazione sul dataset prodotto, successivamente è riportato un confronto con gli elenchi già esistenti e infine vengono elencate le problematiche riscontrate durante la raccolta dei dati.

4.1 Analisi sui dati raccolti

Il dataset prodotto contiene un totale di 1368 Associazioni Non Profit che operano nella provincia di Bologna. Di seguito sono riportati i risultati

ottenuti per mezzo dell'analisi effettuata relativamente ai dati raccolti per ogni associazione:

Informazioni sul luogo 1229 associazioni risultano avere le informazioni relative all'ubicazione della sede, corripondente circa al 90% sul totale.

Contatti telefonici Le associazioni a cui risultano collegati uno o più recapiti telefonici sono 948, ovvero circa il 69%.

Indirizzi email Le associazioni a cui sono associati uno o più indirizzi email sono 955, corssipondente al 70%.

Sito Web 1000 associazioni, ovvero circa il 73%, hanno un Sito Web di riferimento.

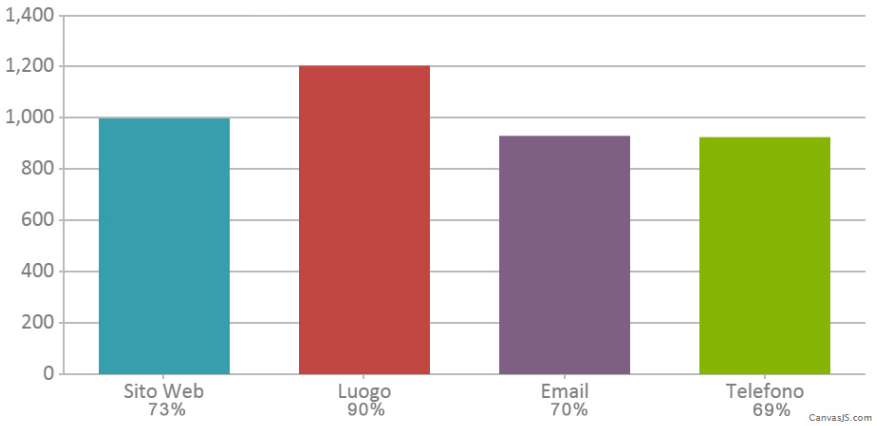

Figura 4.1: Grafico relativo ai dati raccolti (Bologna)

Entrando più nel dettaglio riguardo alle informazioni relative al luogo, il 68% risulta avere l'indirizzo completo (via, CAP, Comune, Regione), mentre il restante 32% contiene solamente la località (quindi Comune, CAP e Regione).

Per quanto riguarda le categorie, le associazioni che appartengono ad una o più categorie oltre ad 'Associazione Non Profit' (categoria di default) sono 798 (58%). La ripartizione delle categorie è visibile nel grafico in figura 4.2 .

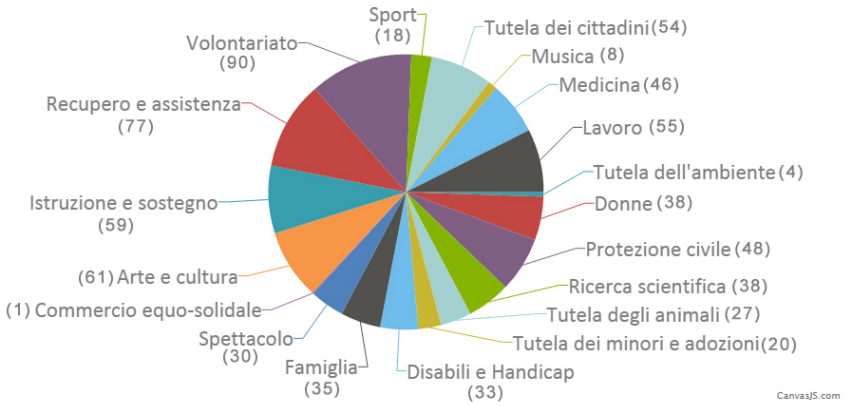

Figura 4.2: Ripartizione delle categorie (Bologna)

La categoria a cui risultano associate più organizzazioni è 'Volontariato', seguita da 'Recupero e assistenza' e 'Istruzione e sostegno'. Le categorie per cui, invece, si hanno i risultati più bassi sono 'Commercio equo-solidale' (a cui appartiene una sola associazione), 'Tutela dell'ambiente' (solo 4 associazioni) e 'Musica' (8 associazioni).

Le associazioni che risultano appartenenti alla sola categoria di default 'Associazione Non Profit') sono 600, circa il 43%.

4.1.1 Dati della regione Emilia-Romagna

Dopo aver ottenuto un buon numero di dati relativi alla provincia di Bologna, si è pensato di iniziare ad espandere il dataset relativamente all'intera regione dell'Emilia-Romagna.

Al momento il dataset risulta composto da un totale di 2570 associazioni; la ripartizione delle associazioni per provincia è riportata nella tabella 4.1 .

Comune	# Associazioni
Bologna	1368
Modena	251
Ferrara	59
Forlì-Cesena	45
Parma	62
Piacenza	38
Ravenna	45
Rimini	41

Tabella 4.1: Numero di associazioni per provincia

I grafici in figura 4.3 e 4.4 mostrano rispettivamente la suddivisione per categoria e il numero di informazioni ottenute per ogni tipologia.

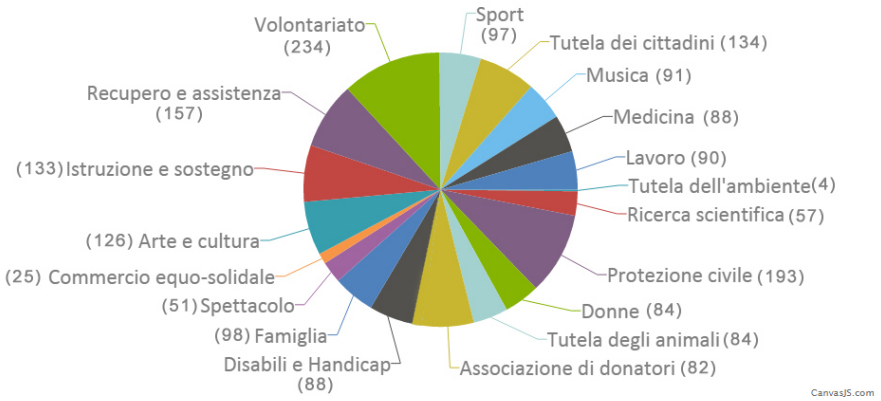

Figura 4.3: Ripartizione delle categorie (Emilia-Romagna)

Come è possibile vedere, le categorie con il maggior numero di associazioni sono 'Volontariato', 'Protezione civile' e 'Recupero e assistenza'; la categoria con il minor numero di associazioni risulta 'Tutela dell'ambiente'.

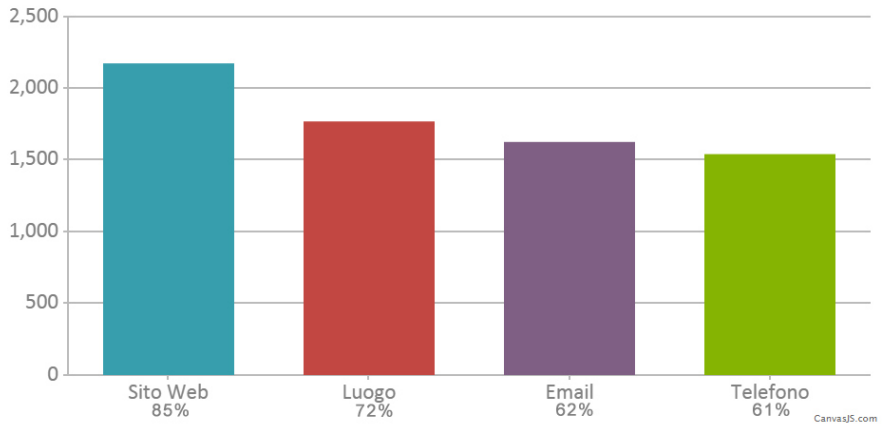

Figura 4.4: Grafico relativo ai dati raccolti (Emilia-Romagna)

L'ampliamento del dataset con i dati relativi all'intera regione non è ovviamente completo e risulta ancora in fase di sviluppo: occorre infatti raccogliere un numero maggiore di dati ed effettuare ulteriori controlli sui dati già ottenuti; questi sono sicuramente i primi obiettivi da portare a termine nell'ambito dei possibili sviluppi futuri.

4.2 Confronto con gli elenchi esistenti

Per poter avere un quadro completo del lavoro svolto, ho effettuato un confronto tra il dataset da me creato sulla provincia di Bologna e gli elenchi già esistenti e raggiungibili mediante i seguenti link:

- https://wwwservizi.regione.emilia-romagna.it/teseofe/associazioni-promozione-sociale.asp
- http://www2.provincia.bologna.it/associaz.nsf/

Per quanto riguarda il primo elenco, per il comune di Bologna sono presenti circa 790 associazioni, mentre il mio dataset ne contiene 1368, ovvero 578

in più che corrisponde in percentuale ad un +78%. Nel mio dataset quindi risultano presenti tutte le associazioni che si trovano nell'elenco indicato.

Il confronto con il secondo elenco è risultato un po' più complicato a causa del modo in cui le informazioni sono pubblicate. Non è mi è stato possibile effettuare un'analisi automatica (ad esempio utilizzando lo scraping) per determinare delle corrispondenze, e inoltre nell'elenco non è presente alcun riferimento riguardo al numero totale di associazioni presenti. Ho effettuato quindi un confronto manuale concentrandomi su una specifica categoria, ovvero 'Volontariato', in quanto si tratta della categoria sulla quale ho più dati a disposizione. Su un confronto effettuato su un campione di 160 associazioni prelevate dall'elenco provinciale, sono emersi i seguenti risultati:

- Per la maggior parte delle associazioni considerate non è indicato un Sito Web di riferimento; nello specifico si tratta di 122 associazioni su 160. Di queste 122 associazioni, 13 sono risultate invece presenti nel mio dataset con l'URL del relativo Sito Web.

- Considerando le rimanenti 38/160 associazioni:

  - 23 risultano presenti anche nel mio elenco;
  - 15 risultano mancanti.

Successivamente ho prelevato un campione di 80 associazioni dal mio dataset e ho effettuato delle ricerche all'interno dell'elenco:

- per 40 associazioni non ho ottenuto alcun risultato;

- le restanti 40 associazioni sono risultate invece presenti nell'elenco.

Secondo questi risultati quindi, il mio dataset contiene il 50% di associazioni in più.

4.3 Problematiche comuni riscontrate durante l'estrazione dei dati

Ho condotto un'analisi su alcune associazioni per le quali non ho ottenuto determinate informazioni, con lo scopo di capire quali fossero i motivi legati a queste mancanze.

Per quanto riguarda le informazioni di contatto, cioè indirizzi email e recapiti telefonici, i casi in cui non ho ottenuto risultati sono i seguenti:

- Le informazioni non erano presenti sul Sito Web;
- Le informazioni relative ai contatti erano inserite in elementi statici della pagina (es. immagini);
- Lo script non è stato in grado di recuperare i link delle pagine contenenti le informazioni;
- Nel sito era presente un form per contattare direttamente l'associazione, quindi gli indirizzi non erano specificati esplicitamente.

I problemi relativi alle informazioni sul luogo sono invece risultati i seguenti:

- Non era presente alcuna informazione riguardo l'ubicazione;
- Le informazioni erano inserite in elementi statici della pagina (es. immagini) oppure riportati tramite l'utilizzo dell'API *Google Maps*;
- Lo script non è stato in grado di recuperare i link delle pagine contenenti le informazioni;
- L'espressione regolare utilizzata non è stata in grado di identificare l'indirizzo, in quanto espresso in una forma non considerata.

Per quanto riguarda le categorie occorre fare un discorso più ampio, che non riguarda solamente la mancanza di dati, ma anche l'effettiva correttezza delle informazioni ottenute. Su un campione di 100 associazioni:

- Il 67% risulta associato alle giuste categorie;
- Il 15% risulta associato alle categorie sbagliate;
- Il restante 18% risulta in parte associato alle categorie corrette e in parte a quelle sbagliate (questo è possibile in quanto ogni associazione può appartenere ad una o più categorie).

Le cause degli errati accostamenti tra un'associazione e le relative categorie sono dovuti principalmente da due fattori, elencati in ordine di rilevanza:

- L'algoritmo di stemming applicato alle parole chiave, in alcuni casi, ha fatto emergere delle corrispondenze tra parole che in realtà non avevano alcun elemento in comune.
- In altri casi, le parole chiave restituite da *AlchemyAPI* non avevano molta attinenza con il contesto e di conseguenza risultavano inutili ai fini dell'analisi.

Per quanto riguarda invece i casi in cui non è stato possibile determinare una categoria, le motivazioni sono sostanzialmente due: non è stata trovata alcuna corrispondenza tra le parole chiave restituite da *AlchemyAPI* e l'elenco di parole chiave da me definito; *AlchemyAPI* non ha restituito nessuna parola chiave associata al Sito Web analizzato.

Conclusioni

Il progetto svolto in questo lavoro di tesi prevedeva due obiettivi principali. Il primo obiettivo era quello di sviluppare un sistema che permettesse di raccogliere le informazioni relative alle Associazioni Non Profit presenti sul Web; il secondo obiettivo consisteva nel sistematizzare i dati raccolti per creare e pubblicare un dataset secondo le specifiche definite dai Linked Open Data.

Il caso d'uso affrontato in questo trattato fa riferimento in particolare al mondo del Non Profit della provincia di Bologna. In questo senso i risultati ottenuti sembrano positivi, in quanto grazie ai tools implementati è stato possibile raccogliere molte informazioni che hanno portato alla creazione di un dataset ricco e ben strutturato.

Uno dei punti di forza dei tools di Crawling e Scraping implementati è che presentano una struttura modulare, che li rende facilmente modificabili e ne favorisce il riutilizzo nella loro interezza o anche solamente in parte. Questo permette di poter adattare tali strumenti per un utilizzo in contesti diversi e più ampi rispetto a quello considerato, al fine di ottenere dati significativi per diversi scopi e realtà di interesse.

Il dataset prodotto, rispettando le direttive dei Linked Open Data e utilizzando come riferimento gli standard definiti dal W3C, rappresenta un'ottima base da cui partire per sviluppare e fornire ulteriori servizi. Inoltre, essendo pubblicati in modo aperto, i dati ottenuti possono essere integrati e ampliati con dati provenienti da altre fonti, in maniera molto semplice.

Occorre tener presente che il lavoro svolto rappresenta solamente il primo

passo verso la realizzazione di un progetto più ampio, e che quindi necessita di essere portato avanti in futuro.

Uno sguardo al futuro

Parlando di sviluppi futuri, gli scenari che si prospettano sono molteplici. Da un lato c'è la possibilità di utilizzare il dataset prodotto per sviluppare dei servizi innovativi e utili alla comunità. L'idea che ha portato alla realizzazione di questo lavoro di tesi era quella di creare un sistema che permettesse di raccogliere e centralizzare le informazioni relative ad eventi, attività e corsi organizzati dalle varie associazioni: il lavoro svolto fornisce gli strumenti necessari per poter procedere in questa direzione. Il passo successivo richiederà di adattare i tools messi a disposizione in modo che siano in grado di estrapolare anche le informazioni relative alle nuove realtà d'interesse.

Per quanto riguarda le funzionalità implementate, restano ancora diversi punti su cui lavorare per aumentarne l'efficienza e creare un sistema più completo. In primo luogo, uno dei punti su cui si può pensare di lavorare nell'immediato futuro è l'integrazione dei Social Network come mezzo tramite il quale reperire le informazioni sulle associazioni; in riferimento a questo, è stata inserita una sezione specifica nella quale viene fornita una panoramica sul come poter inserire i profili Facebook negli algoritmi di ricerca implementati per il progetto.

Altri punti su cui si può pensare di lavorare sono relativi alle problematiche di cui si è parlato nella sezione 4.3; nello specifico occorre ottimizzare gli algoritmi implementati per gestire anche i casi che al momento non sono stati considerati (come ad esempio cercare di recuperare le informazioni relative ai luoghi che sono pubblicate nelle pagine per mezzo dell'API Google Maps). Inoltre, si può pensare di migliorare l'algoritmo relativo all'associazione delle categorie, in particolare cercando di perfezionare la ricerca di corrispondenze tra le parole chiave che allo stato attuale rappresenta uno dei punti deboli

dell'algoritmo prosposto.

Integrazione dei Social Network: Facebook

Al momento i Social Network sono esclusi dall'attività di analisi effettuata: se il Crawler si imbatte in un URL appartenente ad un profilo Social, lo scarta immediatamente. È importante, però, pensare di integrare anche i vari Social Network nella ricerca effettuata, in quanto al giorno d'oggi rappresentano uno dei canali più utilizzati per divulgare e reperire informazioni. In questa sezione si vuole dare un'idea su come poter integrare i profili Facebook all'interno degli algoritmi implementati

Facebook mette a disposizione una serie di API che permettono agli sviluppatori di interagire in modo efficace con il famoso Social Network. L'API che può essere presa in considerazione per raggiungere l'obiettivo che ci interessa è la *Graph API*. L'idea su cui si basa quest'API è quella di considerare elementi quali utenti, pagine e gruppi, come nodi di un grafo che risultano tra loro collegati tramite archi. Ad esempio, dal nodo *user* partono gli archi *friends*, che collegano ciascun utente ad altri utenti (ovvero i suoi amici).

L'API può essere chiamata effettuando una richiesta HTTP all'URL `http://graph.facebook.com/`*`id-della-pagina`*. La chiamata restituisce un oggetto di tipo JSON ed ha un formato simile (sono riportate solo le informazioni che ci interessano):

```
{
"id": "ID PAGINA",
"about": "DESCRIZIONE",
"category": "CATEGORIA DI APPARTENENZA"
...
"link": "URL DELLA PAGINA",
"location": {
  "city": "COMUNE",
  "country": "NAZIONE",
```

```
  "street": "INDIRIZZO",
  "zip": "CAP"
  },
"phone": "NUMERO TELEFONO",
"website": "URL SITO WEB",
...
 }
```

Estrapolare le informazioni di interesse risulterà quindi molto semplice, in quanto l'oggetto restituito contiene già quasi tutto quello che ci occorre. Una possibile implementazione può essere la seguente:

```
1  function findInfoFromFacebook($id_pagina){
2    try{
3      $content = file_get_contents("http://graph.facebook.
           com/".$id_pagina);
4    }
5    catch{
6      $array_content = json_decode($content,true);
7      $associazione = array();
8      /*Esempio di recupero info luogo*/
9      if(array_key_exists("location",$array_content) ===
           true){
10       $associazione['luogo'] = array();
11       $associazione['luogo']['comune'] =  $array_content
             ['location']['city'];
12       $associazione['luogo']['cap'] =  $array_content['
             location']['zip'];
13       $associazione['luogo']['indirizzo'] =
             $array_content['location']['street'];
14       ...
15 }
```

Nel codice è riportato un esempio relativo al recupero delle informazioni riguardo il luogo, ma lo stesso principio può essere seguito per estrapolare

tutte le altre informazioni.

Una volta creato l'oggetto `$associazione` il procedimento da seguire per il salvataggio delle informazioni risulterà analogo a quello già implementato per i normali Siti Web e quindi non sarà richiesto ulteriore codice.

Appendice A

Elenco query SPARQL

Di seguito sono elencate alcune query utili per interrogare lo SPARQL endpoint.

Elenco completo delle associazioni

Verrà restituito un elenco completo delle associazioni, con tutti i relativi dati.

```
1  prefix skos: <http://www.w3.org/2004/02/skos/core#>
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3  prefix org: <http://www.w3.org/ns/org#>
4  prefix vcard: <http://www.w3.org/2006/vcard/ns#>
5  prefix foaf:  <http://xmlns.com/foaf/0.1/>
6
7  SELECT ?nome_associazione ?link ?stato ?regione ?cap
8         ?locality ?indirizzo
9  (GROUP_CONCAT(DISTINCT ?email ; separator= ';') AS
10  ?contatti_email)
11 (GROUP_CONCAT(DISTINCT ?numero ; separator= ';') AS
12  ?contatti_tel)
13 (GROUP_CONCAT(DISTINCT ?purp_label ; separator= ';') AS
14  ?categorie)
15 WHERE{
```

```
16  ?org a org:Organization;
17  skos:prefLabel ?nome_associazione;
18  org:purpose ?purpose.
19  ?purpose rdfs:label ?purp_label.
20  OPTIONAL{ ?org foaf:homepage ?link}
21  OPTIONAL{ ?org org:hasSite ?site.
22          ?site a org:Site;
23          org:siteAddress ?site_address.
24          ?site_address a vcard:Location.
25          OPTIONAL {?site_address vcard:hasAddress ?
               address.
26           ?address a vcard:Work.
27           ?address vcard:country-name ?stato;
28           vcard:region ?regione;
29           vcard:locality ?locality.
30           OPTIONAL{?address vcard:postal-code ?cap}
31           OPTIONAL{?address vcard:street-address ?
                indirizzo.}
32          }
33          OPTIONAL {?site_address vcard:hasEmail ?email}
34          OPTIONAL {?site_address vcard:hasTelephone ?
               telephone.
35           ?telephone a vcard:Voice;
36           vcard:hasValue ?numero
37          }
38  }}
39  GROUP BY ?nome_associazione ?link ?stato ?regione ?cap
40  ?locality ?indirizzo ORDER BY ?nome_associazione
```

Ricerca delle associazioni in base ad una categoria

Verrà restituito l'elenco delle associazioni (e relativi dati) che appartengono alla categoria (o alle categorie) di interesse (in questo esempio 'Arte e

cultura').

```
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix org: <http://www.w3.org/ns/org#>
prefix vcard: <http://www.w3.org/2006/vcard/ns#>
prefix foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?nome_associazione ?link ?stato ?regione ?cap
       ?locality ?indirizzo
(GROUP_CONCAT(DISTINCT ?email ; separator= ';') AS
 ?contatti_email)
(GROUP_CONCAT(DISTINCT ?numero ; separator= ';') AS
 ?contatti_tel)
(GROUP_CONCAT(DISTINCT ?purp_label ; separator= ';') AS
 ?categorie)
WHERE{
?org a org:Organization;
skos:prefLabel ?nome_associazione;
org:purpose ?purpose.
?purpose rdfs:label ?purp_label.
OPTIONAL{ ?org foaf:homepage ?link}
OPTIONAL{ ?org org:hasSite ?site.
       ?site a org:Site;
       org:siteAddress ?site_address.
       ?site_address a vcard:Location.
       OPTIONAL {?site_address vcard:hasAddress ?
          address.
        ?address a vcard:Work.
        ?address vcard:country-name ?stato;
        vcard:region ?regione;
        vcard:locality ?locality.
        OPTIONAL{?address vcard:postal-code ?cap}
        OPTIONAL{?address vcard:street-address ?
```

```
            indirizzo.}
32          }
33          OPTIONAL {?site_address vcard:hasEmail ?email}
34          OPTIONAL {?site_address vcard:hasTelephone ?
               telephone.
35           ?telephone a vcard:Voice;
36           vcard:hasValue ?numero
37          }
38  }
39   FILTER (?purp_label = 'Arte e cultura')
40  }
41   GROUP BY ?nome_associazione ?link ?stato ?regione ?cap
42   ?locality ?indirizzo ORDER BY ?nome_associazione
```

Ricerca delle associazioni in base al comune

Verrà restituito l'elenco delle associazioni (e relativi dati) che hanno sede nel comune (o nei comuni) di interesse (in questo esempio 'Casalecchio di Reno').

```
1   prefix skos: <http://www.w3.org/2004/02/skos/core#>
2   prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3   prefix org: <http://www.w3.org/ns/org#>
4   prefix vcard: <http://www.w3.org/2006/vcard/ns#>
5   prefix foaf:  <http://xmlns.com/foaf/0.1/>
6
7   SELECT ?nome_associazione ?link ?stato ?regione ?cap
8          ?locality ?indirizzo
9   (GROUP_CONCAT(DISTINCT ?email ; separator= ';') AS
10   ?contatti_email)
11  (GROUP_CONCAT(DISTINCT ?numero ; separator= ';') AS
12   ?contatti_tel)
13  (GROUP_CONCAT(DISTINCT ?purp_label ; separator= ';') AS
14   ?categorie)
```

```
15  WHERE{
16  ?org a org:Organization;
17  skos:prefLabel ?nome_associazione;
18  org:purpose ?purpose.
19  ?purpose rdfs:label ?purp_label.
20  OPTIONAL{ ?org foaf:homepage ?link}
21  ?org org:hasSite ?site.
22          ?site a org:Site;
23          org:siteAddress ?site_address.
24          ?site_address a vcard:Location.
25          ?site_address vcard:hasAddress ?address.
26           ?address a vcard:Work.
27           ?address vcard:country-name ?stato;
28           vcard:region ?regione;
29           vcard:locality ?locality.
30           OPTIONAL{?address vcard:postal-code ?cap}
31           OPTIONAL{?address vcard:street-address ?
                indirizzo.}
32          OPTIONAL {?site_address vcard:hasEmail ?email}
33          OPTIONAL {?site_address vcard:hasTelephone ?
               telephone.
34           ?telephone a vcard:Voice;
35           vcard:hasValue ?numero
36          }
37  FILTER ( ?locality = 'Casalecchio di Reno')
38  }
39  GROUP BY ?nome_associazione ?link ?stato ?regione ?cap
40  ?locality ?indirizzo ORDER BY ?nome_associazione
```

Ricerca delle associazioni in base al nome

Verrà restituito l'elenco delle associazioni (e relativi dati) il cui nome contiene il termine specificato (in questo esempio 'Sport').

```
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix org: <http://www.w3.org/ns/org#>
prefix vcard: <http://www.w3.org/2006/vcard/ns#>
prefix foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?nome_associazione ?link ?stato ?regione ?cap
       ?locality ?indirizzo
(GROUP_CONCAT(DISTINCT ?email ; separator= ';') AS
 ?contatti_email)
(GROUP_CONCAT(DISTINCT ?numero ; separator= ';') AS
 ?contatti_tel)
(GROUP_CONCAT(DISTINCT ?purp_label ; separator= ';') AS
 ?categorie)
WHERE{
?org a org:Organization;
skos:prefLabel ?nome_associazione;
org:purpose ?purpose.
?purpose rdfs:label ?purp_label.
OPTIONAL{ ?org foaf:homepage ?link}
OPTIONAL{ ?org org:hasSite ?site.
       ?site a org:Site;
       org:siteAddress ?site_address.
       ?site_address a vcard:Location.
       OPTIONAL {?site_address vcard:hasAddress ?
          address.
        ?address a vcard:Work.
        ?address vcard:country-name ?stato;
        vcard:region ?regione;
        vcard:locality ?locality.
        OPTIONAL{?address vcard:postal-code ?cap}
        OPTIONAL{?address vcard:street-address ?
           indirizzo.}
```

```
32          }
33          OPTIONAL {?site_address vcard:hasEmail ?email}
34          OPTIONAL {?site_address vcard:hasTelephone ?
               telephone.
35           ?telephone a vcard:Voice;
36           vcard:hasValue ?numero
37          }
38 }
39  FILTER regex(?nome_associazione, 'Sport', 'i')
40 }
41  GROUP BY ?nome_associazione ?link ?stato ?regione ?cap
42  ?locality ?indirizzo ORDER BY ?nome_associazione
```

Elenco completo delle categorie

Verrà restituito l'elenco completo delle categorie disponibili.

```
1  prefix skos: <http://www.w3.org/2004/02/skos/core#>
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3  prefix org: <http://www.w3.org/ns/org#>
4  prefix vcard: <http://www.w3.org/2006/vcard/ns#>
5  prefix foaf:  <http://xmlns.com/foaf/0.1/>
6
7  SELECT ?categoria ?descrizione
8  WHERE{
9  ?categoria a skos:Concept;
10 rdfs:label ?descrizione.
11 } ORDER BY ?descrizione
```

Bibliografia

[1] ISTAT, 'La rilevazione sulle istituzioni Non Profit: Un settore in crescita' (2013). http://www.istat.it/it/files/2013/07/05-Scheda-Non-Profit_DEF.pdf.

[2] Nigam A., 'Web Crawling Algorithms', *International Journal of Computer Science and Artificial Intelligence* 4 (2014): 63-67.

[3] Srivastav G.K., Ali I., Srivastava K.A., 'Overview of Search Engine and Crawler', *International Conference of Advance Research and Innovation* (2014): 217-219.

[4] Sharma S., Kumar R., 'Web-Crawling Approaches in Search Engines', Marzo 2008.

[5] Patel A., Patel M., 'A survey on information retieval from Web using Web Scraping technique', *International Journal of Innovative Research in Technology (IJIRT)* 1 (2014):95-100.

[6] Open Knowledge Foundation, 'Open Data Definition' 2.0 (ultima visita: Gennaio 2015). http://opendefinition.org/od/.

[7] Obama B., 'Memorandum for the Heads of Executive Departments and Agencies' (Marzo 2009). http://www.whitehouse.gov/the_press_office/TransparencyandOpenGovernment.

[8] Data.gov, 'Open Data Internationally' (ultima visita: Gennaio 2015). http://www.data.gov/open-gov/.

[9] Vignati P.M., 'Introduzione al Semantic Web', *MokaByte 116* (Marzo 2007).

[10] Signore O., 'RDF per la rappresentazione della conoscenza' (2003). http://www.w3c.it/papers/RDF.pdf.

[11] Bernes-Lee T., 'Linked data-design issues' (2006). http://www.w3.org/DesignIssues/LinkedData.html.

[12] Dati.gov.it, 'Infografica' (Ultima visita: Gennaio 2015). http://www.dati.gov.it/content/infografica.

[13] Wikipedia, 'Struttura della Partita IVA'. http://it.wikipedia.org/wiki/Partita_IVA#Struttura_della_partita_IVA.

[14] Open Data Comune di Bologna, 'Che cos'è l'Open Data Index' (Marzo 2012). http://dati.comune.bologna.it/node/154.

[15] Gruber T. R., 'A translation approach to portable ontologies', *Knowledge Acquisition* 5(2) (1993): 199-220. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[16] Google, 'Mission 501(c)(3): Driving Donations, Digitally' (Settembre 2013). http://think.withgoogle.com/databoard/media/pdfs/digital-non-profits-study_research-studies_1.pdf.

[17] Poste italiane, 'Standard di composizione di indirizzi' (Aprile 2009). http://www.poste.it/resources/editoriali/postali/pdf/cap_standard_composizione_indirizzi.pdf.

Ringraziamenti

Innanzitutto voglio ringraziare i miei genitori, per aver sempre creduto in me e nelle mie capacità, incoraggiandomi a non smettere mai di inseguire i miei sogni e spronandomi a fare sempre del mio meglio; se sono riuscita a raggiungere questo importante traguardo lo devo soprattutto a loro e all'immancabile sostegno che mi hanno sempre offerto in tutto e per tutto.

Voglio poi ringraziare mio fratello Nicolò, perchè da bravo fratello maggiore mi è sempre stato vicino, mi ha aiutato a crescere ed è sempre stato per me un esempio da seguire e un punto di riferimento a cui non potrei mai rinunciare. Ringrazio inoltre mia cognata Marica per avermi fatto uno dei regali più belli che la vita potesse offrirmi, ovvero la mia stupenda nipotina Giada!

Ringrazio zii, zie, cugini e cugine per avermi sempre incoraggiato ed essere sempre stati presenti in ogni circostanza.

Voglio ringraziare le mie nonne, per essere state una presenza fondamentale della mia vita sin da quando ero piccola ed in particolare per tutto l'affetto che mi hanno sempre dimostrato.

Un ringraziamento speciale lo voglio fare anche ai miei nonni, perchè nonostante non abbiano potuto essermi fisicamente accanto in questo percorso so che in un modo o nell'altro sono sempre stati vicino a me come lo saranno per tutta la vita.

E parlando di famiglia, non può mancare un ringraziamento alla mia sorella col pelo, Shila, perchè è proprio vero che *'chi non ha mai posseduto un cane non può sapere cosa significhi essere amato'*.

Ringrazio i miei amici più cari, Valentina, Alessia, Alice, Maurizio e Ombretta per tutte le giornate e le avventure passate insieme, ma soprattutto per essermi sempre stati vicini e avermi sostenuto come solo i veri amici sanno fare; grazie a loro ho capito il vero significato dell'Amicizia.

Ringrazio anche tutti i 'ragazzi della collinetta' e i loro compagni a 4 zampe per tutti i bei pomeriggi trascorsi insieme e per il bellissimo rapporto che si è creato tra noi, da un anno a questa parte.

Ci tengo inoltre a ringraziare anche tutti gli amici, più o meno stretti, che hanno fatto parte della mia vita e che lo sono tutt'ora, grazie per tutte le serate, le cene, le uscite al cinema e le risate che mi avete regalato in questi anni. Ringrazio inoltre tutti i colleghi e compagni universitari che hanno condiviso con me le gioie e i dolori di questi anni e con i quali ho affrontato esami, progetti, tirocini e interminabili sessioni di studio.

Infine, voglio ringraziare il Prof. Angelo Di iorio, relatore, e il Dott. Francesco Poggi, co-relatore, per tutto l'aiuto e il sostegno fornitomi durante la realizzazione di questa tesi.