Wikipedia Category Map

Laboratorio di Intelligenza Artificiale e Robotica.

Bandera Carlo (711484)

Indice

Capitolo 1

Introduzione

Wikipedia Category Map è un progetto che si inserisce nella vastità di lavori riguardanti il Semantic-Web. Questa branca della computer science si pone come obiettivo l'organizzazione dei contenuti nel web aggiungendo informazioni che permettano un'interpretazione degli stessi.
In una prima e sommaria visione Wikipedia può essere vista come un insieme di pagine (chiamate Articoli). Queste pagine vengono organizzate in categorie. Ogni categoria può avere associate delle pagine e delle sotto-categorie a formare un albero.
In Figura 1.1 è mostrata la pagina *http://it.wikipedia.org/Categoria:Film*.
Le pagine in Wikipedia vengono scritte dagli utenti stessi dell'enciclopedia. Ad ogni pagina viene associata una categoria o un insieme di categorie a discrezione degli utenti. I collegamenti tra pagine e categorie sono quindi non vincolati, autoregolati dalla comunità di utenti che frequentano Wikipedia.
Il problema di questo approccio è che gli utenti non riescono ad avere una visione completa delle informazioni che modificano. Può darsi ad esempio che associando una pagina ad una categoria, a causa dell'organizzazione gerarchica delle categorie, si rischi di avere un'inconsistenza lungo l'albero. In

Figura 1.1: Pagina Categoria:Film

particolare è molto probabile che l'albero possa diventare un grafo presentando in alcuni casi una organizzazione della conoscenza che non rispecchi la normale concezione reale.

Lo scopo di questo progetto è fornire uno strumento utile agli ingegneri della conoscenza che permetta loro di evidenziare, dove possibile, eventuali inconsistenze in modo da poterle correggere.

Il tool renderà possibile poi lo studio della struttura gerarchica delle categorie consentendo un controllo di correttezza semantica delle relazioni.

Capitolo 2

Risorse e librerie utilizzate

2.1 MediaWiki (versione 1.15.0)

MediaWiki è un software wiki libero scritto in PHP. Il suo utilizzo è legato in generale a Wikipedia ma può essere usato con tutti i siti "wiki". Mediawiki mette a disposizione un'interfaccia attraverso la quale è possibile facilmente interrogare il wiki per ottenere informazioni sui suoi contenuti.
All'indirizzo http://en.wikipedia.org/w/api.php si possono trovare tutte le funzionalità offerte con una esaustiva documentazione su come utilizzarle.
In generale l'utilizzo è molto semplice. Per ottenere un'interazione basta postporre all'indirizzo del wiki che si vuole interrogare una serie di parametri con specificati opportuni valori.
Ad esempio action=login serve per autenticarsi sul sito web e ricevere il token a seguito del buon esito dell'operazione. La lista di parametri disponibile per questa azione è:

- lgname - Username
- lgpassword - Password

- lgdomain - Indica il dominio. Questo parametro è opzionale.

Quindi ad esempio questa URL:
http://it.wikipedia.org/w/api.php?action=login&lgname=user&lgpassword=password
permette l'autenticazione sul sito http://it.wikipedia.org/ con la username *user* e la password *password*.
Fino a questo punto dello sviluppo del progetto l'utilizzo di MediaWiki è molto limitato. In particolare Wikipedia viene interrogata con una query per estrarre informazioni su una categoria specificata.
Per fare ciò si utilizza *list=categorymembers* che permette di ottenere tutte le pagine direttamente legate ad una categoria.
Sono supportati i seguenti parametri:

- cmtitle - la categoria della quale vogliamo ottenere informazioni.
- cmprop - che informazioni a riguardo includere nella risposta: ids, title, sortkey, timestamp. I valori devono essere separati dal carattere "|".
- cmnamespace - includere solo le pagine nel set di namespace indicato. I valori disponibili sono: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100 e 101. Il separatore è sempre il carattere "|". I namespace sono molto importanti in quanto permettono di distinguere i vari elementi nella risposta. Ad esempio se un elemento ha come namespace 14 significa che rappresenta una categoria, se ha come namespace 0 vuol dire invece che si tratta di una pagina.
- cmcontinue - per le categorie grandi indica da quale valore partire a produrre il risultato
- cmlimit - indica il limite massimo di elementi nella risposta. Di default è 10 e al massimo può essere 500.

- cmsort - serve per ordinare gli elementi della risposta. I valori possono essere sortkey (valore di default) e timestamp.
- cmdir - serve per indicare se ordinare in modo ascendente o discendente.
- cmstart cmend - indicano il timestamp dal quale partire e finire.
- cmstartsortkey cmendsortkey - indicano la sortkey (carattere alfanumerico) dalla quale partire e finire.

Per esempio la query:
http://en.wikipedia.org/w/api.php?
action=query&list=categorymembers&cmtitle=Category:Physics
serve per ottenere le prime dieci pagine della Category:Physics.
Il risultato può essere ottenuto sotto forma di XML:

```
<?xml version="1.0"?>
<api>
<query>
    <categorymembers>
      <cm pageid="22939" ns="0" title="Physics"/>
      <cm pageid="24489" ns="0" title="Outline of physics"/>
      <cm pageid="1653925" ns="100" title="Portal:Physics"/>
      <cm pageid="3445246" ns="0" title="Glossary of classical physics"/>
      <cm pageid="22950086" ns="0" title="Guide to Semiconductor Physics"/>
      <cm pageid="2129107" ns="14" title="Category:Fundamental physics concepts"/>
      <cm pageid="4769321" ns="0" title="Timeline of fundamental physics"/>
      <cm pageid="694942" ns="14" title="Category:Physicists"/>
      <cm pageid="1198" ns="0" title="Acoustics"/>
      <cm pageid="1913676" ns="14" title="Category:interdisciplinary physics"/>
    </categorymembers>
  </query>
  <query-continue>
    <categorymembers cmcontinue="Atomic, molecular, and optical physics"/>
  </query-continue>

</api>
```

2.2 Wikiont

Per poter attribuire un significato semantico ai contenuti nel web c'è bisogno di fissare un'ontologia di riferimento. Così facendo si può assegnare ad ogni elemento un URI che permetta in maniera automatica di associare ad esso l'interpretazione corretta.

In rete sono presenti innumerevoli ontologie costruite per descrivere i vari domini applicativi. Ce ne sono che descrivono le transazioni economiche, la struttura delle società, le funzionalità di web server e molte altre. Per quanto riguarda Wikipedia, esiste un'ontologia che ne descrive il contenuto: *Wikiont*[1]. L'ontologia classifica le pagine di Wikipedia in vari termini (articoli, categorie, immagini, autore...). Wikiont è stata sviluppata dal DERI[2] ed è scritta in OWL[3].

L'unico elemento che, per l'obiettivo del progetto risulta mancante, è una relazione che leghi gli articoli alle categorie. Per questo è stato deciso di aggiungere all'ontologia la proprietà http://purl.org/dc/elements/1.1/subject[4] che permette di colmare la mancanza. Infine per ragioni applicative è stato deciso di portare l'intera ontologia in RDF. Per fare ciò è stata usata la libreria JENA che ha permesso una facile traduzione.

In Figura 2.1 è mostrata l'ontologia di riferimento.

In questo progetto verranno usate le seguenti classi:

- Article - http://airwiki.elet.polimi.it/rdf/wikiont.rdf♯Article
- Category - http://airwiki.elet.polimi.it/rdf/wikiont.rdf♯Category

e le seguenti relazioni:

[1] http://semanticweb.org/wiki/WikiOnt

[2] http://www.deri.ie/

[3] uno tra i più noti linguaggi per ontologie

[4] resa disponibile dalla nota ontologia dublin core

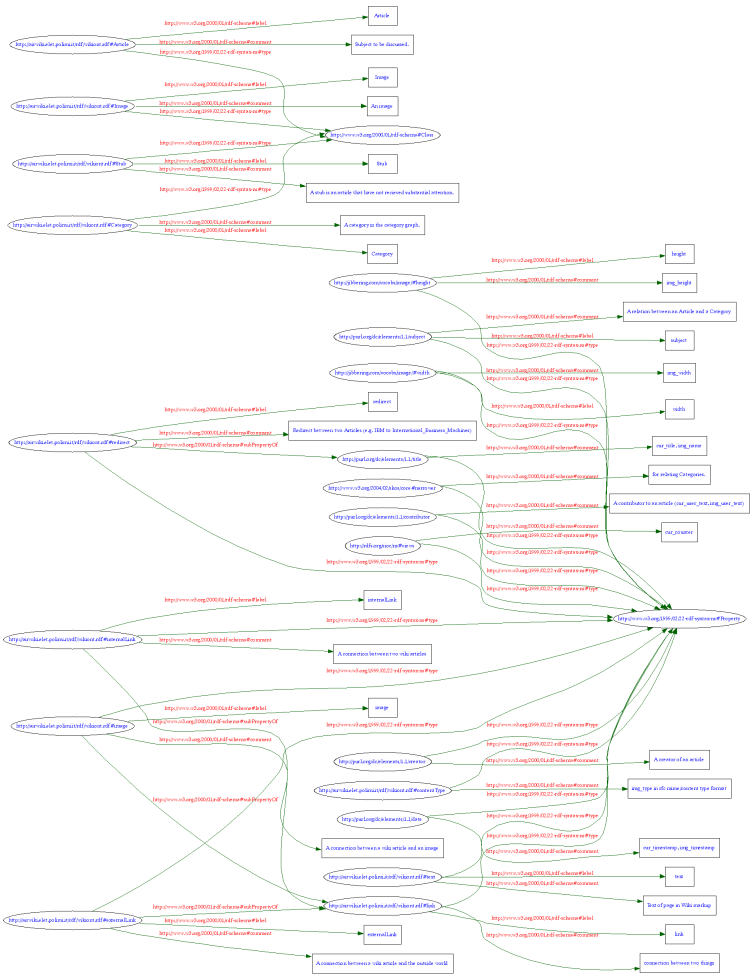

Figura 2.1: L'ontologia Wikiont

- Subject - che come già scritto lega le pagine alla categoria alla quale sono direttamente collegate http://purl.org/dc/elements/1.1/subject.
- Narrower - indica relazioni gerarchiche tra categorie http://www.w3.org/2004/02/skos/core♯narrower.

2.3 JENA (versione 2.6.0)

Jena è un framework Java che permette di creare applicazioni per il Semantic Web. Fornisce un ambiente di programmazione per RDF, RDFS, OWL, SPARQL e include un motore di inferenza rule-bases.
Sul sito web del progetto http://jena.sourceforge.net/index.html nella sezione downloads è possibile ottenere una copia della libreria. Essa può essere importata direttamente tra le librerie dell'applicazione che si sta sviluppando e subito utilizzata senza bisogno di alcuna installazione.
Verrà ora affrontata la parte relativa alla creazione e all'utilizzo di file RDF in modo da poter meglio capire l'implementazione del progetto.
Brevemente, l'RDF (Resource Description Framework) è uno standard W3C per descrivere *risorse*. In concetto di risorsa è molto generico e può inglobare qualsiasi cosa (la homepage di un sito web, una persona, un libro ecc.).
Ogni risorsa viene identificata attraverso una URI e può avere delle *proprietà* associate. Nella visione a grafo utilizzata nella rappresentazione di RDF, i nodi del grafo rappresentano le risorse mentre gli archi tra i nodi rappresentano le proprietà. Quest'ultime a loro volta si contraddistinguono attraverso una URI. Oltre ad avere un identificativo, la proprietà, essendo a sua volta una risorsa, può avere proprietà associate.
È possibile creare dei tipi di risorse. Ad esempio un rdfs:Literal è una sottoclasse di rdfs:Resource ed è un'istanza di rdfs:Class. rdfs:Literal è definita come la classe che rappresenta una sequenza di caratteri.
Per facilità di rappresentazione, le risorse possono avere un namespace associato. Esso non è altro che il prefisso del nome compresi i due punti (":"). Ogni namespace usato deve essere definito all'inizio del file RDF. Riprendendo l'esempio sopra, la risorsa Literal (rdfs:Literal) con il namespace definito come xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema♯, per

sostituzione, risulta aver associata la URI http://www.w3.org/2000/01/rdf-schema♯Literal.

L'utilizzo dei namespace in definitiva consente l'utilizzo delle URI abbreviate.

Vediamo ora come i file e gli elementi di RDF vengono trattati in Jena.

Sempre in riferimento all'ontologia sviluppata (Wikiont) supponiamo di avere un articolo (es. http://it.wikipedia.org/Tom_Watson) e una categoria (es. http://it.wikipedia.org/Categoria:Golfisti_statunitensi). Vogliamo esprimere questa relazione in RDF utilizzando Jena.

Come prima cosa è necessario ottenere un modello RDF che rappresenti il file:

```
Model m = ModelFactory.createDefaultModel();
```

Successivamente per comodità vengono definiti dei namespace:

```
m.setNsPrefix(dc, http://purl.org/dc/elements/1.1/);
m.setNsPrefix(wiki, http://it.wikipedia.org/);
m.setNsPrefix(airlab, http://airwiki.elet.polimi.it/rdf/wikiont.rdf#);
```

Creo la proprietà e le due risorse che mi servono nel modello:

```
Property p = m.createProperty(http://purl.org/dc/elements/1.1/subject);
Resource tom = m.createResource(http://it.wikipedia.org/Tom_Watson);
Resource golf = m.createResource(http://it.wikipedia.org/
                                 Categoria:Golfisti_statunitensi);
```

Aggiungo al modello gli asserti usando il metodo add.

Definisco Golfisti statunitensi come una airlab:Category.

```
Resource cat=m.createResource(http://airwiki.elet.polimi.it/
        rdf/wikiont.rdf#Category);
m.add(golf,RDF.type,cat);
```

Definisco la pagina Tom_Watson come un airlab:Article.

```
Resource art=m.createResource(http://airwiki.elet.polimi.it/
        rdf/wikiont.rdf#Article);
m.add(tom,RDF.type,art);
```

Definisco la relazione tra Tom Watson e la categoria al quale è associato.

```
m.add(tom,p,golf);
```

Posso a questo punto scrivere il modello creato. Per la scrittura può essere utilizzato qualsiasi output stream. In questo caso viene usato il System.out.

```
m.write(System.out,"RDF/XML-ABBREV");
```

Che produce il seguente risultato:

```
<rdf:RDF
    xmlns:airlab="http://airwiki.elet.polimi.it/rdf/wikiont.rdf#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:wiki="http://it.wikipedia.org/">
  <airlab:Article rdf:about="http://it.wikipedia.org/Tom_Watson">
    <dc:subject>
      <airlab:Category rdf:about="http://it.wikipedia.org/Categoria:Golfisti_statunitensi"/>
    </dc:subject>
  </airlab:Article>
</rdf:RDF>
```

2.4 JUNG (versione 2.0)

JUNG (Java Universal Network/Graph Framework http://jung.sourceforge.net/) è una libreria software che fornisce strumenti per la modellizzazione, l'analisi e la visualizzazione di dati che possono essere rappresentati come un grafo o una rete.
È stata scelta questa libreria vista la flessibilità e la completezza delle soluzioni disponibili.
Di seguito viene mostrato come sia possibile creare grafi e rappresentarli utlizzando JUNG.
Graph$\langle V, E \rangle$ è un'interfaccia presente nel package edu.uci.ics.jung.graph che rappresenta un generico grafo. V e E sono due tipi generici Java che rappresentano i tipi dei nodi del grafo (V) e degli archi o lati (E).
Ad esempio con Graph⟨String, Number ⟩ si definisce un grafo nel quale gli identificativi dei nodi sono delle stringhe mentre quelli dei lati sono numeri.
L'interfaccia Graph espone metodi per:

- aggiungere e rimuovere lati e vertici dal grafo e ottenere l'insieme di tutti i lati e vertici del grafo.
- ottenere informazioni sugli *endpoints* dato l'identificativo di un lato.
- ottenere informazioni sui nodi come il grado, i predecessori e i successori.

Esistono poi tre sotto-interfacce di Graph che identificano tipi diversi di grafo: *DirectedGraph*, *UndirectedGraph* e *SimpleGraph*.
Il codice a seguire mostra la creazione di un semplice grafo.

```
// Graph<V, E> V rappresenta il tipo dei vertici
// E rappresenta il tipo degli archi.
```

```
Graph<Integer, String> g = new SparseMultigraph<Integer, String>();
// Aggiungo dei vertici. Per definizione devono essere interi.
g.addVertex((Integer)1);
g.addVertex((Integer)2);
g.addVertex((Integer)3);
// Aggiungo degli archi. Per definizione devono essere stringhe.
// Non possibile aggiungere archi con lo stesso identificativo.
g.addEdge("Edge-A", 1, 2);
g.addEdge("Edge-B", 2, 3);
// Mostriamo in console quello che abbiamo.
System.out.println("Il grafo g = " + g.toString());
```

Il risultato in console è:

Il grafo g = Vertices:1,2,3
Edges:Edge-B[2,3] Edge-A[1,2]

Vengono ora introdotti i principali elementi per la visualizzazione del grafo. Per una completa trattazione si rimanda al sito web di riferimento della libreria.

I componenti essenziali per la visualizzazione di un grafo sono:

- Il grafo. Costruito come mostrato sopra.
- Un'implementazione di un *Layout*. Un layout definisce il modo con il quale i nodi saranno disposti. Nell'esempio che segue sarà usato CircleLayout.
- Un *BasicVisualizationServer*. Il componente swing nel quale verrà visualizzato il grafo.
- Un componente GUI come JFrame dove posizionare il VisualizationServer.

Il codice che segue mostra un utilizzo semplice di queste componenti:

```
public static void main(String[] args) {
    Graph g = ...// Il grafo creato prima
    // Il layout parametrizzato come il grafo
    Layout<Integer, String> layout = new CircleLayout(g);
        // inizializza la dimensione iniziale
    layout.setSize(new Dimension(300,300));
     // BasicVisualizationServer<V,E>
     BasicVisualizationServer<Integer,String> vv =
              new BasicVisualizationServer<Integer,String>(layout);
         //inizializzazione della dimensione iniziale
     vv.setPreferredSize(new Dimension(350,350));

     JFrame frame = new JFrame("Simple Graph");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.getContentPane().add(vv);
     frame.pack();
     frame.setVisible(true);
}
```

Il risultato è mostrato in Figura 2.2

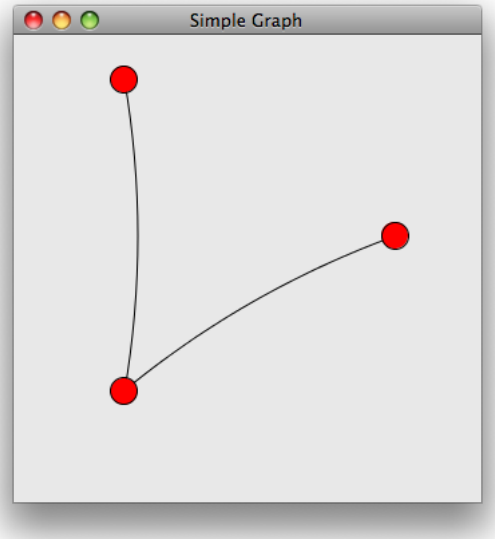

Figura 2.2: Esempio di visualizzazione di un grafo

Capitolo 3

Architettura del sistema

In Figura 3.1 sono mostrati i package dell'applicazione.
L'organizzazione è la classica struttura delle applicazioni Multi-tier a più livelli. È stato deciso di adottare questo pattern in modo da avere già una struttura del sistema che consenta di essere facilmente estesa in futuro, fino a poter supportare l'interfacciamento col web e databeses.

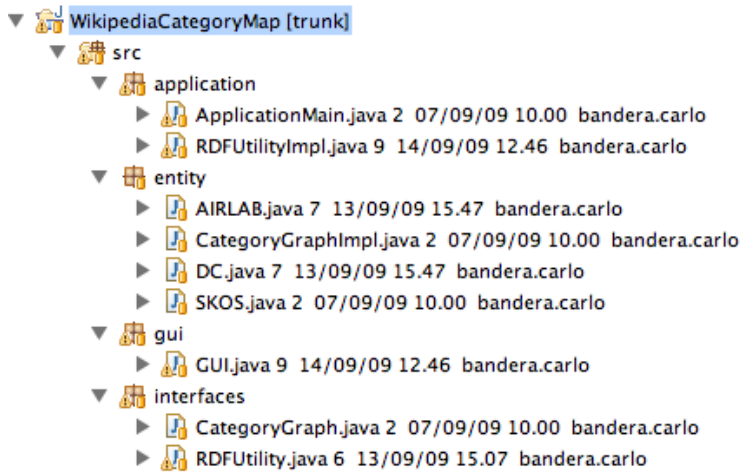

Figura 3.1: Organizzazione dei package

```
public interface RDFUtility {

    public String generaRDF(String addr, String categoria);

    public CategoryGraph generaGrafo(String rdf, String addr, String categoria);

    public Vector<String> getPagesOfCategory(String rdf, String addr, String categoria,
            boolean recursive);

}
```

Figura 3.2: RDFutility.java

L'organizzazione è la seguente:

- Package *interfaces*. Qui sono definite due interfacce: *RDFUtility.java* e *CategoryGraph.java*.
  La prima definisce l'oggetto che si occuperà di interfacciarsi con la libreria per lo sviluppo di ontologie (JENA in questa implementazione). Ogni classe che userà questa interfaccia, dovrà implementare i seguenti metodi (Figura 3.2):
  - *generaRDF* prende in ingresso come parametri l'indirizzo di un wiki basato su MediaWiki (ad esempio *http://it.wikipedia.org/*), la categoria dalla quale si vuole cominciare l'analisi dei contenuti (ad esempio *Categoria:Matematica* nel caso della Wikipedia italiana) e in base alle ontologie di riferimento genera l'RDF.
  - *generaGrafo* dato un file RDF come primo parametro, l'indirizzo di wiki come secondo e come terzo la categoria, genera una rappresentazione a grafo dell'RDF in base alla categoria indicata.
  - getPagesOfCategory prende in ingresso quattro parametri. I primi tre hanno lo stesso significato del metodo generaGrafo mentre l'ultimo parametro indica se le pagine che si vogliono ottenere sono solo quelle direttamente legate alla categoria indicata o se si vuo-

le percorre il sotto-grafo generato da quella categoria e ottenere tutte le pagine direttamente legate alle categorie del sotto-grafo.

La seconda definisce la struttura dati utilizzata dall'applicazione per rappresentare le relazioni tra categorie. In particolare si limita a fissare le operazioni disponibili sulla struttura dati che sono *public void add(String padre, String figlio);* che serve per aggiungere un arco tra due nodi (cioè una relazione gerarchica tra categorie) e *public Object getGraph();* che permette di ottenere il grafo. Qui il valore di ritorno è stato lasciato generico in modo da poter restituire ciò che si crede meglio in fase di implementazione.

- Package *Entity*. Il nome del package deriva da *Entity Object*. In questo package infatti ci sono le classi che rappresentano il businnes model. C'è l'implementazione di CategoryGraph e delle classi che servono a definire il vocabolario utilizzato per la rappresentazione in RDF. Queste classi contengono esclusivamente una serie di attributi statici che definiscono le URI delle risorse. Ad esempio se nel codice ci si riferisce a DC.subject, vuol dire che ci si sta riferendo alla relazione definita da http://purl.org/dc/elements/1.1/subject.
  Più precisamente fino a questo punto dell'implementazione del progetto le proprietà e le classi usate nel modello (sempre in riferimento all'ontologia definita in http://airwiki.elet.polimi.it/rdf/wikiont.rdf) sono: *AIRLAB.category* e *AIRLAB.article* per le classi, *SKOS.narrower* per le relazioni tra categorie e *DC.subject* per le relazioni Articoli-Categorie. In futuro l'utilizzo di altre risorse comporterà l'inserimento delle URI negli opportuni vocabolari (cioè nelle classi associate).
  Per quanto riguarda l'implementazione del grafo si è deciso di rappre-

sentarlo come un Vector⟨Vector⟨String⟩⟩. Il vettore esterno contiene un'insieme di espansioni. Per espansioni si intende una serie di stringhe (vettori interni) nella quale in prima posizione è inserito il "padre" mentre a seguire vengono elencati i "figli" se ce ne sono.

- Package *Application*. Qui sono state inserite le classi che riguardano la businnes logic. *ApplicationMain.java* è semplicemente la classe responsabile dell'avvio del software mentre *RDFUtilityImpl.java* è l'implementazione dell'interfaccia RDFUtility. In questa classe vengono utilizzate le api di MediaWiki per l'estrazione dei contenuti da un wiki arricchiti poi da una descrizione semantica grazie alla libreria JENA. Una volta ottenuto l'RDF, la generazione del grafo e il recupero delle pagine data una categoria sono stati facilmente ottenuti grazie all'utilizzo di query SPARQL messe a disposizione dalla libreria.

- Package *GUI*. Questo package contiene tutto ciò che riguarda l'interfaccia grafica. La classe GUI si interfaccia con la RDFUtilityImpl grazie alla sua interfaccia. In ApplicationMain viene prima creata un'istanza della classe RDFUtilityImpl e poi un'istanza di GUI passandogli come argomento l'istanza di RDFUtilityImpl che per GUI è un oggetto conforme all'interfaccia RDFUtility.

  La classe GUI è una classe che contiene molte componenti. Nel prossimo capitolo verrà analizzato il contenuto di questa classe commentando esaustivamente le singole operazioni che espone. Per la visualizzazione del grafo questa classe si appoggia alla libreria JUNG vista la completezza, la facilità di utilizzo e flessibilità nella rappresentazione dei grafi.

L'architettura del progetto è stata pensata in modo da tenere al massimo

disaccoppiate le varie componenti che compongono il sistema. L'interfacciamento e l'utilizzo di RDF avviene solo nelle classi presenti nel package application. Le entità che rappresentano il modello e che, ad esempio, potranno essere mappate su un database, sono solo nel package Entity. Infine la parte di visualizzazione che coinvolge la libreria JUNG è contenuta solo nel package GUI.
Così facendo, se si dovesse rendere necessario un cambio di libreria (JUNG o JENA), solo alcuni package ne saranno coinvolti riuscendo così a diminuire gli sforzi di migrazione.

Capitolo 4

Guida all'utilizzo del software

Questo capitolo serve da guida all'utilizzo di *Wikipedia Category Map*.
In Figura:4.1 è mostrata la finestra principale del programma. Nella parte superiore, a partire da sinistra, si trovano:

- Il campo indirizzo. In questo spazio è possibile inserire l'indirizzo del wiki che si desidera. In figura è riportato quello di Wikipedia in italiano. Il campo è obbligatorio.

- Il campo categoria. Lì va indicato qual è la categoria dalla quale si vuole cominciare l'analisi delle pagine e categorie di Wikipedia. Il campo naturalmente non può rimanere vuoto. Esso deve contenere la parola *Categoria:* (nel caso si stia interrogando la Wikipedia italiana) seguito dal nome della categoria "radice".
  Il nome della categoria deve seguire una sintassi ben precisa. Più precisamente il nome della categoria deve rispettare le seguenti regole:

  – Se il nome contiene degli spazi, devono essere sostituiti con degli underscore (" _ "). Es: *Filosofia della matematica* diventa *Filosofia_della_matematica*.

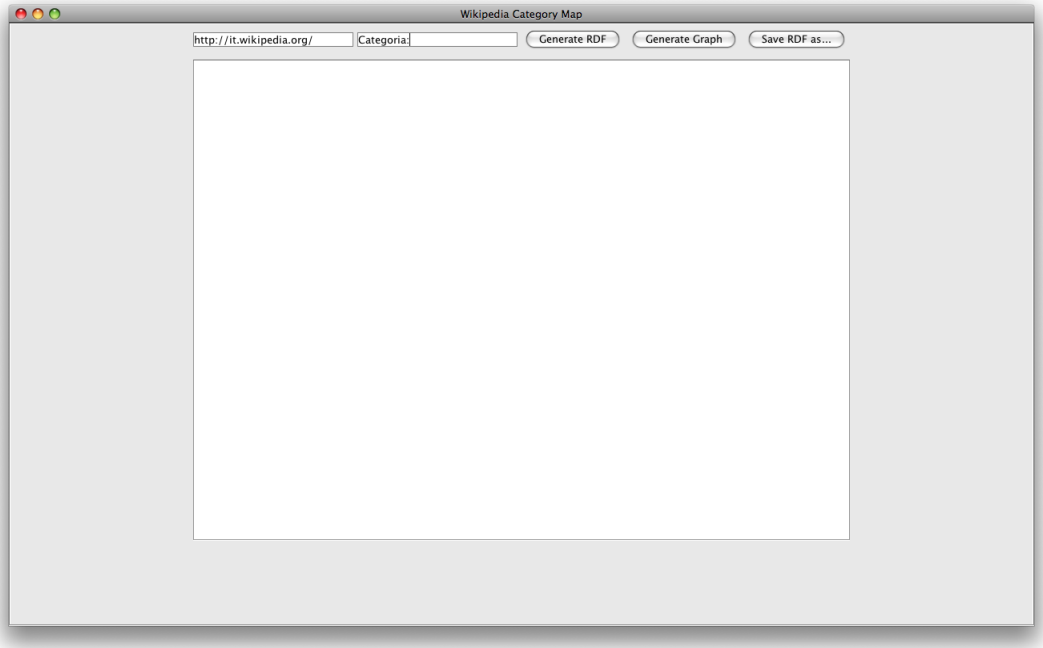

Figura 4.1: Finestra principale

  - Se il nome contiene degli apostrofi, devono essere sostituiti con %27. Es: *Patrimoni dell'umanità in Italia* diventa *Patrimoni_dell%27umanità_in_Italia*.
  - Se il nome contiene il simbolo %, deve essere sostituito da %25.

- Il pulsante *Generate RDF*. Una volta completati i campi a sinistra, questo pulsante genera l'RDF delle pagine e categorie e lo visualizza nello spazio sottostante. Nella text area è possibile (mediante copia e incolla) caricare un RDF già conforme a Wikiont. In questo caso, va indicata la categoria "radice" nell'apposito campo.

- Il pulsante *Generate graph*. Questo pulsante serve per visualizzare sottoforma di grafo, le categorie e le loro relazioni. Questo pulsante aprirà una finestra che verrà spiegata in seguito.

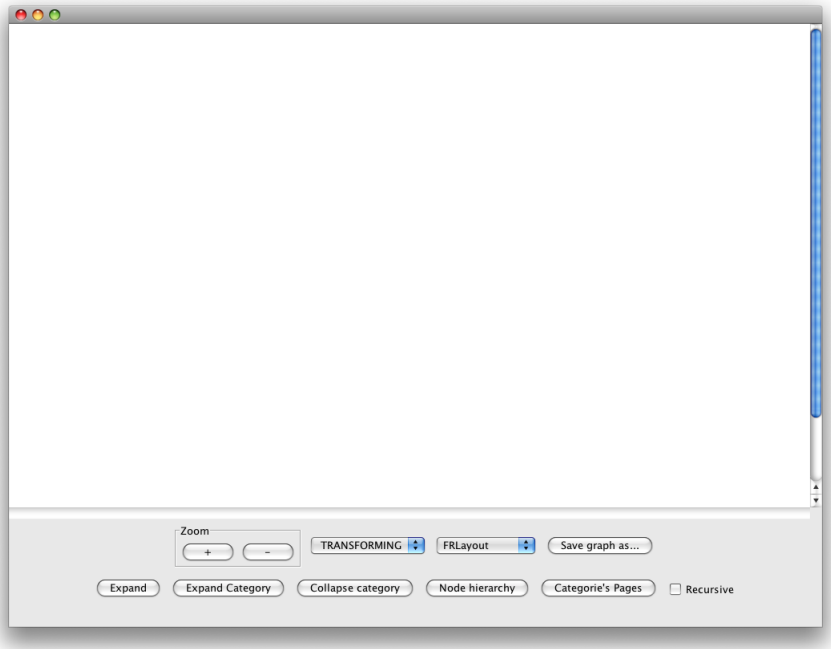

Figura 4.2: Finestra che conterrà il grafo

- Il pulsante *Save RDF as...*. Questo pulsante permette di salvare il contenuto dell'area di testo che si trova in centro alla finestra.

Le interrogazioni a Wikipedia possono richiedere molto tempo. Infatti l'implementazione del sistema prevede che venga effettuata una query ogni 500ms. Quindi, più generica è la categoria, più elementi contiene e più tempo ci impiegherà il programma a recuperare tutte le informazioni.

In Figura 4 è mostrata la finestra dove viene visualizzato il grafo. La parte centrale è un pannello che conterrà il grafo. Nella parte della finestra in basso ci sono i pulsanti che permettono di controllarlo.
Le azioni disponibili sono:

- Zoom. I due pulsanti + e - servono per ingrandire o rimpicciolire

rispettivamente le dimensioni del grafo. Questa operazione può anche essere fatta comodamente usando la scrolling wheel del mouse.

- Modo di interazione. Il combo box a fianco serve per scegliere la modalità di intervento sul grafo. Se è selezionata la modalità TRANSFORMING è possibile spostare l'intero grafo tenendo premuto il pulsante sinistro (per i destrorsi) del mouse. La modalità PICKING permette di selezionare e spostare i singoli elementi del grafo o un insieme di essi.

- Scelta del layout. A priori non è possibile scegliere un layout per il grafo che vada bene in generale. Per questo è stato introdotto un combo box che permette di selezionare i layout messi a disposizione dalla libreria in modo da poter selezionare quello più adeguato al grafo visualizzato.

- *Save Graph as....* Questo pulsante permette di salvare uno screenshot della parte di grafo visualizzata nel pannello. Il file avrà l'estensione e il formato *.png*

- *Expand.* Questo pulsante permette di espandere passo passo i nodi del grafo.

- *Expand Cateogory.* Il pulsante permette di espandere una categoria in particolare. Si sceglie la modalità PICKING dal combo box, si sceglie una categoria (è supportata anche la selezione multipla, per ottenerla è necessario premere il tasto *shift*) e si schiaccia il pulsante.

- *Collapse Category.* In caso non si fosse interessati all'ulteriore espansione di una categoria o di una parte del grafo e possibile selezionarla e premere il pulsante Collapse Category. È supportata la selezione multipla di categorie. Una volta chiusa una categoria, può essere riaperta semplicemente utilizzando il pulsante Expand Category.

- *Node Hierarchy.* Questo pulsante permette l'isolamento di parte del grafo. Si può isolare solo una categoria o più categorie in caso si voglia scoprire la relazione tra esse. Questo pulsante apre una nuova finestra. Nella nuova finestra viene data la possibilità di intervenire sul grafo con la modalità PICKING oppure di salvare il sotto-grafo ottenuto in un file.

- *Category's Page.* Con questo pulsante è possibile vedere le pagine associate alla categoria selezionata. È possibile anche qui selezionare più categorie. In tal caso verrà visualizzata un'unione insiemistica delle pagine. La lista di pagine verrà visualizzata in una finestra separata, dando anche la possibilità di salvarla in un file.

- *Recursive.* Questa check box permette di ottenere le pagine non solo della categoria selezionata, ma di tutto il sotto grafo generato da essa.