Politecnico di Milano
V Facoltà di Ingegneria

Corso di laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

GENERAZIONE DI SPANNING TREE A PARTIRE DA GRAFI RDF

Relatore: Prof.ssa Letizia Tanca
Correlatore: Prof.ssa. Mirjana Mazuran

Tesina di Laurea Magistrale di:
Alessio Giulio Feliciano, matricola 765557
Richard Manga Dooh, matricola 765588

Anno Accademico 2012-2013

Un profondo ringraziamento alla professoressa Tanca e Mazuran

per il loro sostegno e disponibilità durante il compimento di questo lavoro.

Ringraziamo, inoltre, il professore Cremonesi per averci seguito durante

Il nostro percorso formativo all'estero.

Il ringraziamento più grande va alle nostre famiglie che nel corso di questi

anni hanno sempre incoraggiato e ci hanno aiutato in tutte le nostre imprese.

Sommario

Tavola delle Figure

Abstract

Il formato RDF è stato concepito come un modello di dati metadata (informazione che descrive un insieme di dati). Negli ultimi anni, il formato RDF si è affermato come un linguaggio di descrizione e di modellizzazione di risorse web.

I file RDF possono quindi essere utilizzati per la descrizione concettuale o la modellizzazione di risorse web memorizzate in posti diversi. Il formato RDF viene utilizzato per generare una descrizione di qualsiasi risorsa, ed è comprensibile da un computer. In effetti, il vocabolario RDF può essere esteso con un numero arbitrario di vocabolario esterno, ciò che facilita la descrizione di qualsiasi risorsa.

L'utilizzazione diffusa del formato RDF e l'estendibilità del suo vocabolario rende dunque questo modello di gestione dei dati un elemento importante per tutti coloro che si interessano al mondo del web.

L'obiettivo di questo lavoro è di fornire uno strumento software che permette di estrarre il collegamento ottimale tra le differenti risorse che compongono il documento RDF.

La rappresentazione del modello RDF sarà fatta tramite un grafo. Ritrovare il collegamento ottimale tra tali risorse vuol dire trovare l'albero di supporto a partire dal grafo. Per raggiungere tale obiettivo sarà impiegato l'algoritmo di Spanning Tree (1). Nel secondo capitolo, presenteremo lo stato dell'arte relativo alla relazione esistente tra dati RDF e dati relazionali.

Nel terzo capitolo, verranno presentati i differenti algoritmi che si possono applicare sui grafi, e che saranno di utilità nel resto della trattazione e per l'applicazione al nostro caso di studio.

Nel quarto capitolo di questo lavoro ,definiremo il formato RDF e presenteremo le caratteristiche principali di questo formato di dati.

Nell'ultimo capitolo, verrà spiegato dettagliatamente l'approccio che è stato usato per raggiungere gli obiettivi fissati.

Introduzione

Il funzionamento del web oggi si basa principalmente sul web browsing umano e sulla ricerca attraverso dei contenuti testuali. Questo modo di funzionamento sta diventando meno adeguato, visto il volume di dati sempre crescente che dobbiamo gestire.

Oggi, quello che si cerca di realizzare nel mondo del web è trovare un modo che garantisca un accesso integrato ed uniforme a tutte le sorgenti di informazione ed a tutti i servizi web.

Un modello del genere richiede meccanismi standard per scambiare dati e gestire eventuali problemi legati alle differenze semantiche tra dati.

Il formato RDF rappresenta un passo importante in questa direzione. I documenti RDF forniscono un modello di dati che permette di realizzare un'integrazione veloce di dati provenienti da sorgenti diverse, colmando le differenze semantiche. Questo tipo di documento viene usato spesso per descrivere dei metadata (informazione che descrive un insieme di dati) che contengono informazioni su altre risorse web, come dei file XML. È importante precisare che rappresentare dati con i metadata è equivalente a rappresentare semplicemente i dati. I documenti RDF possono quindi essere usati come una struttura generica per lo scambio di dati sul web.

Spesso, è interessante capire i diversi pattern che esistono tra i dati o le risorse che sono memorizzati nei diversi formati (RDF, XML…), per ottimizzare la gestione dei contenuti sul web. Per ottenere quest'informazione molto importante, vengono usate delle tecniche di Web mining (applicazione di tecniche di data mining per trovare i pattern). Il Web mining può essere diviso in tre tipi differenti:

- **Web usage mining:** è il processo che consiste ad estrarre l'informazione utile a partire dai log dei server. Un esempio molto intuitivo del Web usage mining può essere la ricerca dei contenuti ai quali gli utenti di Internet sono interessati. Questa tecnica di data mining permette di capire meglio i bisogni degli utenti, e di sviluppare delle applicazioni più adatte ai loro bisogni. Il Web usage mining può a sua volta essere diviso in tre tipi distinti:
    - **Web server data:** si tratta dell'analisi dei log degli utenti raccolti dai web server. I dati tipici che si cercano sono l'indirizzo IP, l'orario di connessione ed il riferimento alla pagina.
    - **Application server data:** gli application server commerciali hanno la caratteristica di permettere la realizzazione di applicazioni di e-Commerce con uno sforzo minimo.

    - **Application Level Data:** nuovi tipi di eventi possono essere definiti in un'applicazione, e si memorizzano tutti gli accessi a tale applicazione.
- **Web structure mining:** si tratta di un processo che consiste ad analizzare la struttura dei nodi e delle connessioni tra i nodi in un sito web. Questa tecnica si basa esclusivamente sulla teoria dei grafi. Ci sono due tipi di Web structure mining:
    - **Estrazione di pattern a partire da hyperlink nel le web:** un hyperlink è un componente strutturale che connette una pagina web ad un'altra locazione.
    - **Mining della struttura del documento:** analisi della struttura ad albero delle pagine per descrivere l'uso delle etichette HTML or XML.
- **Web content mining:** si tratta di fare il mining, l'estrazione e l'integrazione dei dati di interesse a partire dal contenuto di una pagina web.

Questo lavoro è un contributo al Web structure mining. In effetti, dato che il formato dei dati RDF si sta diffondendo per le ragioni viste all'inizio di questo capitolo, diventa opportuno pensare ad un meccanismo che permette di ottimizzare la navigazione tra le varie risorse. Come verrà illustrato in seguito, esiste una relazione stretta tra dati RDF e i grafi (le risorse possono essere viste come nodi, ed gli statement come archi che collegano i nodi). Il contributo di questo lavoro sarà la realizzazione di uno strumento software per ottimizzare l'accesso alle differenti risorse RDF. Per ottimizzare la navigazione a partire da una struttura in forma di grafo, spesso, si usano degli algoritmi che permettono di trovare un albero a costo minimo a partire dal grafo.

Obiettivi del lavoro realizzato.

Il presente lavoro ha come obiettivo di trovare un insieme di collegamenti tra le differenti risorse RDF che minimizzano il numero di salti/link/click per andare da una risorsa <i> ad una risorsa <j>. Per raggiungere tale obiettivo, verrà in un primo momento fatta la trasformazione di documenti RDF in grafi, poi verrà applicato un algoritmo sul grafo ottenuto per trovare il profilo di navigazione ottimale tra le risorse.

Nel capitolo due di questo lavoro, parleremo dello stato dell'arte relativamente ai diversi lavori fatti sui dati RDF. Nei capitoli successivi, verranno illustrati gli strumenti usati per poter realizzare questo lavoro.

Si tratterà principalmente di spiegare i formati di dati RDF (usato per la gestione delle risorse) e XML (usato per il salvataggio del percorso ottimale di navigazione tra le risorse). La conoscenza di questi due formati di dati sarà fondamentale per la realizzazione del nostro strumento software.

Nel quinto capitolo, verrà presentato dettagliatamente il lavoro che è stato realizzato per l'implementazione del nostro strumento di Web structure mining. Si tratterà quindi di descrivere le classi principali, librerie e risultati ottenuti durante il nostro lavoro.

Nell'ultimo capitolo, verrà fatta una conclusione generale del lavoro che è stato svolto.

Stato dell'arte

L'obiettivo principale del Semantic Web è di esporre la vastissima quantità di dati contenuta nelle basi di dati relazionali. Questo obiettivo viene raggiunto realizzando un "Grafo Globale Gigante". Tuttavia, semplici algoritmi per associare in modo automatico i dati relazionali all'RDF sprecano risorse computazionali e sono inefficienti. Questa associazione può quindi essere realizzata usando uno schema delle risorse che viene tagliato per produrre il documento RDF. Il grafo più piccolo risultante produrrà uno strumento potente di query.

Esistono quindi numerosi strumenti che permettono di passare da dati relazionali a risorse RDF. In questo capitolo, presenteremo alcuni degli strumenti che permettono di fare questo compito. Questa associazione tra dati relazionali e dati RDF permette quindi di rendere i dati disponibili per gli strumenti di Semantic Web, e quindi di poter migliorare la gestione dei dati sul web.

Conversione tra dati relazionali e documenti RDF

La conversione di ogni tupla della base di dati relazionale in una tripletta intorno ad un nodo bianco (nodo vuoto) è il metodo adottato nell'articolo (2) ed è rappresentato schematicamente nella figura 1. Il tipo del nodo (implementato come una proprietà rdf:type) è una classe (rdfs:Class) che corrisponde al nome della tabella. Questa trasformazione viene quindi chiamata "Table to Class".

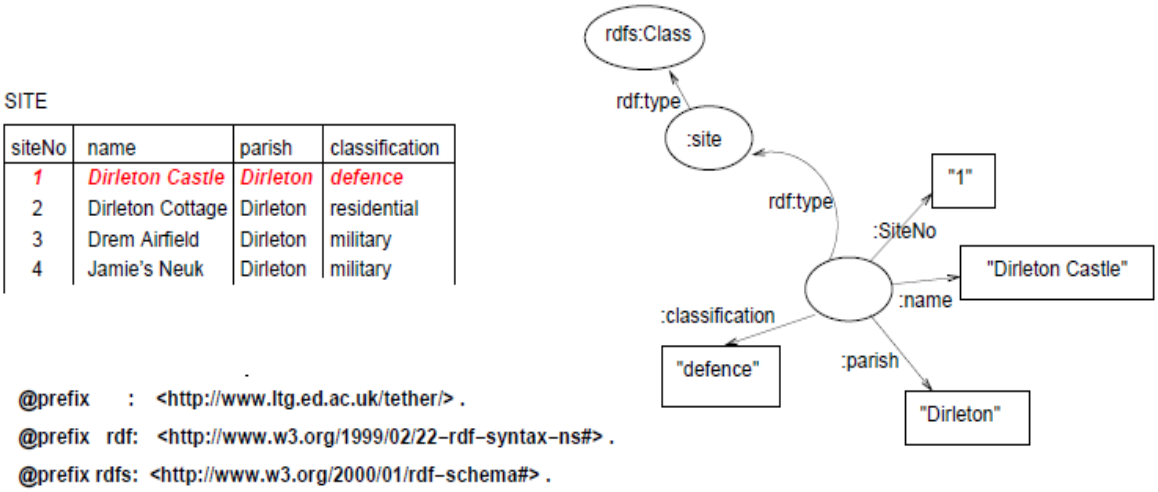

Figura 1 : Trasformazione basica di tuple di una base di dati relazionale in dati RDF

Nell'esempio precedente, si può notare che l'uso di nodi bianchi (vuoti) rende un po' complessa la rappresentazione grafica che è realizzata. Per strutturare meglio questa rappresentazione, è stata realizzata una versione più intuitiva dell'associazione tra basi di dati relazionali e documenti RDF. La figura seguente illustra questa versione che non usa i nodi bianchi.

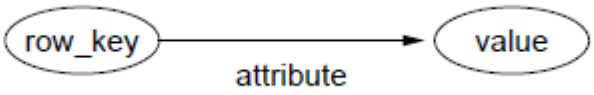

Figura 2 : Corrispondenza tra una tupla e una tripletta RDF.

Nella figura precedente, l'attributo è un URI che deriva direttamente da un campo della base di dati o dal nome di una colonna, ed il valore è il contenuto di questo campo e viene rappresentato come un letterale.

Per quanto riguarda la parola "row_key", viene utilizzata per evitare ogni confusione con la chiave primaria che incontriamo nella maggior parte delle basi di dati relazionali. Questa chiave sarà definita nel dizionario RDF.

La procedura di associazione appena illustrata corrisponde ad un approccio che consiste a fare corrispondere gli attributi della base di dati ad archi del grafo RDF con una precisa conoscenza

dell'origine dei dati. Esistono numerosi strumenti automatizzati che funzionano secondo il principio appena descritto. Alcuni strumenti noti sono: D2RQ, Dartgrid, Dan Connolly's dbview program, R_2O, Triplify. Questi differenti strumenti presentano diversi livelli di personalizzazione per l'utilizzatore.

Esistono anche strumenti che permettono di visualizzare il grafo corrispondente alla base di dati relazionale di cui si fa l'associazione. SquirrelRDF e R2D2 sono degli strumenti che danno la possibilità di vedere il grafo generato da una base di dati relazionale. Con questi due strumenti, si può fare delle interrogazioni SPARQL invece di fare delle interrogazioni SQL. D2RQ e Virtuoso sono degli strumenti che hanno un'opzione per scegliere di istanziare o no i grafi.

Una delle problematiche importanti è di generare degli URI adeguati per le risorse RDF. In effetti, il soggetto e la proprietà di una tripletta RDF devono essere risorse che contengono degli URI, mentre l'oggetto della tripletta può essere un letterale o una risorsa. Se l'oggetto è una stringa letterale, sarà rappresentato nel grafo come una foglia, visto che non potrà essere un soggetto. Come detto precedentemente, ricordiamo che i soggetti i le proprietà devono essere delle risorse.

Scelta degli URI

Uno degli elementi fondamentali per i documenti RDF è che gli URI devono essere differenti quando si riferiscono ad oggetti differenti. Da un altro lato, non ci devono essere una pletora di URI nei documenti RDF, perché questo renderebbe la lettura di tali documenti più pesante. L'approccio usato è quindi di associare alle risorse degli URI canonici, ciò che rende molto semplice il fatto di capire che due URI si riferiscono alla stessa risorsa. Quindi, quando un valore è presente in diversi campi della base di dati, l'URI associato a tutte le occorrenze del valore sarà sempre lo stesso. Ciò che permette di ridurre il numero di URI distinti che vengono usati.

Uso dei nomi, classi, verbi e proprietà

Può essere utile vedere i documenti RDF come une tripletta "soggetto – verbo - oggetto". Quindi, gli archi del grafo diventano un insieme di verbi ed i nodi diventano dei nomi. Il modo migliore per descrivere il tipo di elementi contenuti nella base di dati relazionale è la definizione di classi. Per

quanto riguarda la proprietà dell'arco, è una frase che contiene un verbo che descrive la relazione attiva tra gli elementi.

Uso delle risorse

A prima vista, può essere intuitivo pensare che è meglio usare i letterali invece di usare le risorse. Tuttavia, questo modo di pensare non è giusto, perché i letterali non possono essere soggetti di altre triplette. Ogni risorsa usata dovrebbe quindi avere un tipo associato, in modo da poter sapere la sua caratteristica nella base di dati.

Tuttavia, si potrebbe pensare che tutti i valori della base di dati dovrebbero essere risorse. Bisognerebbe trovare un compromesso, perché vanno trasformati in risorse gli elementi legati al mondo reale o di maggiore importanza per lo studio effettuato.

Gestione delle chiavi primarie

In generale, le chiavi primarie sono generate come numeri. Tuttavia, non si possono unire le chiavi di elementi che hanno significato diverso visto che appartengono a tabelle diverse. Le chiavi primarie hanno quindi sempre un prefisso nei documenti RDF, ciò che permettono di distinguere i valori delle chiavi primarie associate alle differenti tabelle.

Interrogazione dei grafi

I dati RDF ottenuti dopo l'associazione tra basi di dati relazionali e la sintassi RDF sono memorizzati in triplette. Questi dati sono interrogabili tramite l'uso del linguaggio SPARQL.

Diversi magazzini di dati esistono oggi, che permettono di memorizzare tali documenti. I più usati sono Jena (libreria che verrà usata anche per il parsing) e AllegroGraph.

L'unico problema che incontrano gli utilizzatori non esperti è che per fare delle interrogazioni pertinenti, bisognerebbe conoscere bene la terminologia del dominio.

Memorizzazione di dati RDF in basi di dati relazionali

Come visto all'inizio di questo capitolo, i dati in formato RDF sono di particolare importanza per il mondo della Semantic Web. Oggi, è quindi importante poter memorizzare questi dati nelle basi di dati relazionali, che sono quelle più diffuse. Dall'altro lato, interrogare le basi di dati RDF non è così semplice che interrogare le basi di dati relazionali. Un altro fattore molto importante è che gli strumenti di Business Intelligenza supportano principalmente le basi di dati relazionali.

Architettura del sistema

Per trasformare i documenti RDF in una base di dati relazionale, i documenti devono essere caricati in un motore di trasformazione RDF (3). Questo motore è costituito di tre livelli. Le informazioni contenute nello schema RDF saranno usate per costruire la struttura delle tabelle RDF, mentre i dati saranno introdotti nelle tabelle.

La figura seguente rappresenta l'architettura appena descritta.

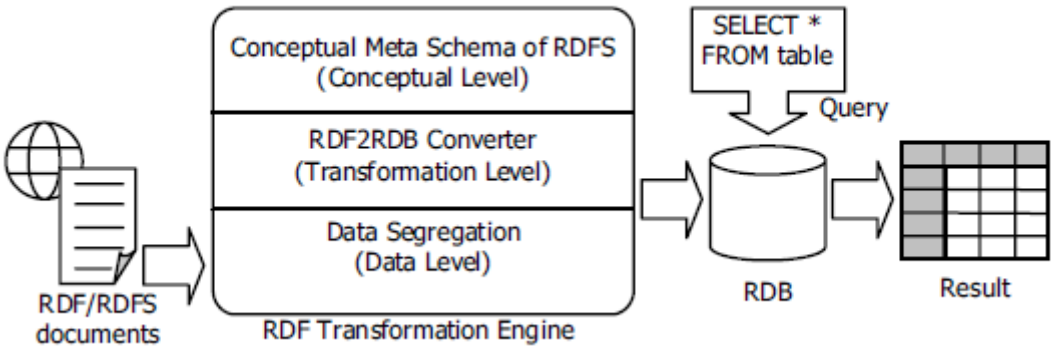

Figura 3 : Architettura del sistema

Il flusso delle operazioni eseguite per ottenere i dati relazionali è descritto nella figura seguente:

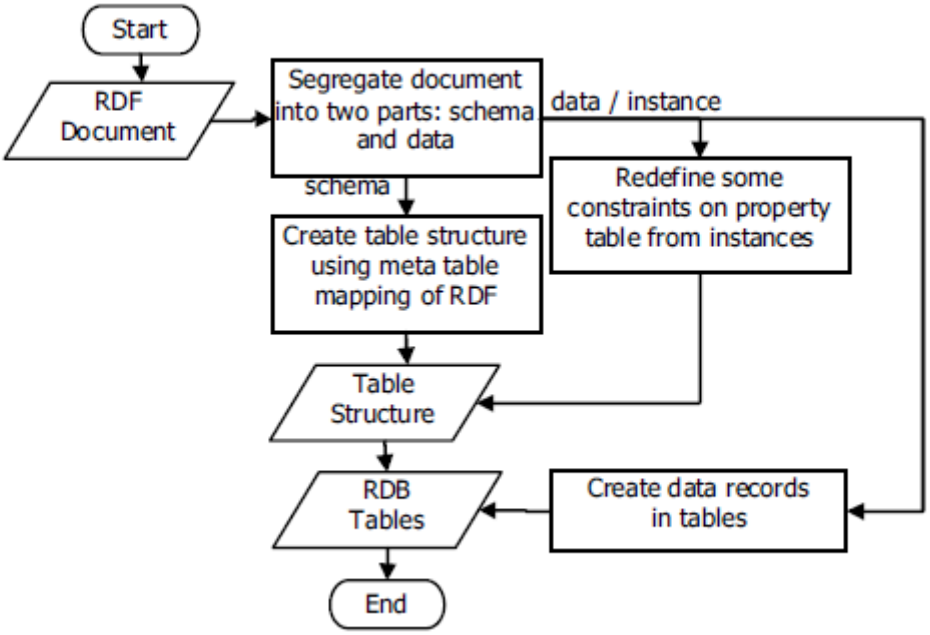

Figura 4 : Flusso delle operazioni per ottenere un modello relazionale dei dati.

Schemi concettuali risultanti da documenti RDF

Per ottenere il modello relazionale, è necessario fare una corrispondenza tra gli elementi dello schema RDF ed i differenti elementi delle basi di dati relazionali. Per fare tale compito, esiste un metodo chiamato NIAM (Nijssen's Information Analysis Method) che permette di realizzare un'associazione tra gli elementi dei due modelli.

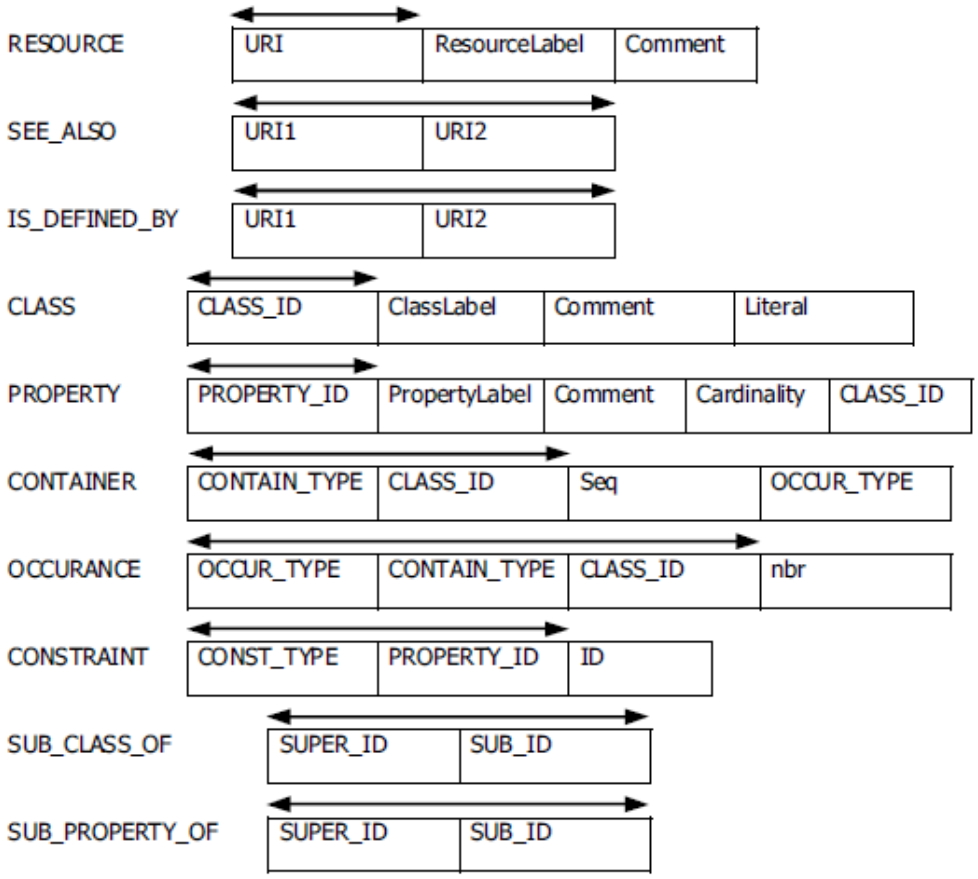

Figura 5 : Associazione tra elementi dello schema RDF e le relazioni.

In questo capitolo, abbiamo visto che esistono tanti strumenti e metodi che permettono di trattare o generare documenti RDF oggi. Tuttavia, una sfida importante rimane il trattamento che si può effettuare su tali dati, per estrarre delle informazioni pertinenti per la gestione delle risorse memorizzate.

Nel prossimo capitolo, verranno descritti i principali algoritmi applicabili sui grafi per estrarne delle informazioni utili.

Algoritmi

Definizione

I grafi sono strutture matematiche discrete che rivestono un grande interesse, sia dal punto di vista matematico, sia per il grande numero di domini d'applicazione. Essi permettono di schematizzare un'ampia varietà di situazioni e di processi.

Un grafo è un insieme di elementi, chiamati nodi, collegati fra loro da archi. La definizione matematica è riportata di seguito:

Definizione : Si dice grafo una coppia $G = (V, E)$ di insieme dove $V = \{v_1, \ldots, v_n\}$ è un insieme finito di elementi detti nodi, mentre $E = \{e_1, \ldots, e_m\} \subseteq V \times V$ è un sottoinsieme di coppie di nodi detti archi o spigoli

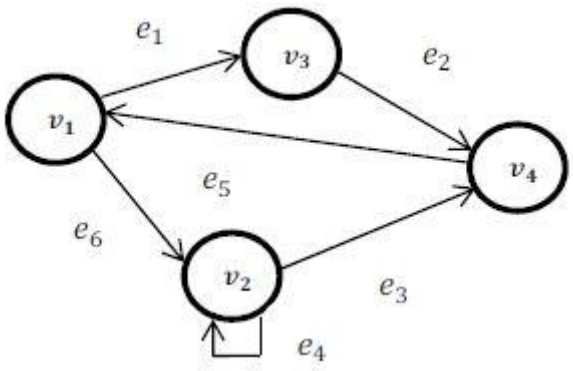

Figura 6 : Esempio di grafo

Ogni nodo è rappresentato come un cerchio, mentre le frecce rappresentano gli archi. Due spigoli si dicono adiacenti se hanno un nodo in comune. Due nodi $v_i \neq v_j$ sono adiacenti se esiste lo spigolo (v_i, v_j).

Alcune proprietà utili per il seguito

Definizione: Un grafo si dice orientato o diretto se ogni arco è una coppia ordinata di vertici $(v_i,v_j) \neq (v_j,v_i)$, cioè l'arco stabilisce una connessione tra il nodo v_i e v_j, ma non viceversa. Se per ogni coppia di nodi esiste il anche l'arco che stabilisce il legame inverso allora il grafo si dice non orientato.

Un'altra proprietà importate dei grafi è la connettività:

Definizione

Un grafo si dice connesso se esiste una sequenza di archi che collega ciascuna coppia di nodi.

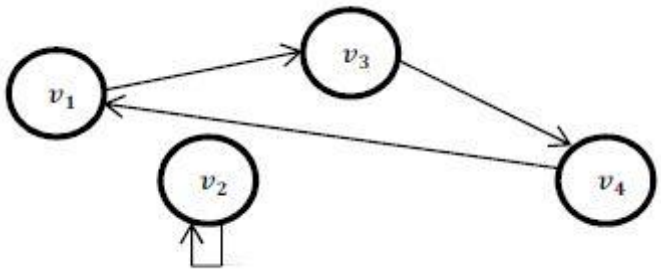

Figura 7 : Grafo no connesso

Nel nostro studio possiamo arrivare ad ottenere dei grafi non connessi a seguito dell'applicazione di un filtro preliminare sui dati. Questo filtro può potenzialmente eliminare degli archi e creare delle partizioni.

Matrice d'adiacenza

Una rappresentazione alternativa per modellare un grafo è la matrice di adiacenza.

Definizione: Sia G = (V, E) un grafo. La di adiacenza del grafo è una matrice quadrata A, di ordine n, dove n = |V| tale che:

$$A_{i,j} = 0 \text{ se } (v_i, v_j) \notin E$$

$$A_{i,j} = 1 \text{ se } (v_i, v_j) \in E$$

Questa modellazione permette di operare sui grafi in maniera più efficace dal punto di vista informatico. E' la rappresentazione più utilizzata al momento della stesura di algoritmi ed è la stessa che verrà utilizzata per implementare l'algoritmo di Page Ranking. La rappresentazione tramite la matrice d'adiacenza, dal punto di vista dell'informazione, è equivalente alla rappresentazione grafica sotto forma di archi e nodi.

Riportiamo un esempio di trasformazione da grafo a matrice di adiacenza.

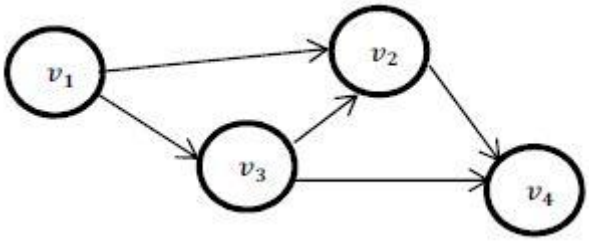

Figura 8 : Grafo corrispondente alla matrice d'adiacenza

Dal grafo di Figura 8 otteniamo che V = {1, 2, 3, 4} e che E = {(1,2); (1,3); (2,4); (3,2); (3,4)} e la matrice di adiacenza

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Notiamo che se il grafo fosse stato orientato la matrice d'adiacenza sarebbe stata simmetrica.

Algoritmo di Page Rank

Cenni storici

Fin dagli albori di internet la ricerca e la catalogazione delle risorse è stato uno dei quesiti che diverse aziende e sviluppatori si sono posti. Nel corso della storia di Internet diversi algoritmi sono stati proposti e utilizzati per portare a termine questo obiettivo. Gli algoritmi si dividono principalmente in due categorie: da un lato gli algoritmi che ricercano nelle pagine i contenuti più pertinenti basandosi sulle parole chiavi, e gli algoritmi che sfruttano le "relazioni" tra le pagine per determinare un ranking. La prima categoria implica una ricerca lessicale delle parole più importanti tramite keyword. Esempio ne è l'algoritmo del portale Altervista. La seconda categoria, di cui fa parte l'algoritmo di Page Rank (4) (5), si basa sul un concetto di notazione delle pagine in funzione dei link entranti e uscenti sulle stesse.

Gli inventori dell'algoritmo di Page Rank, e fondatori di Google, sono Larry Page e Sergey Brin. L'idea di fondo dell'algoritmo è l'ordinamento gerarchico delle risorse che si trovano sul web. Questo ordinamento è dato dall'importanza che la pagina ha sul web. La misura di importanza è basata sul concetto di numero di link entranti in una pagina.

L'algoritmo di Page Rank data di fine anni '90, più precisamente 1997.

Metodo delle potenze (power method)

Il metodo delle potenze è un metodo iterativo per il calcolo dell'autovalore di modulo massimo e il corrispondente autovettore associato all'autovalore. Il seguente metodo riceve in ingresso una matrice quadrata. La parte introduttiva sulla teoria dei grafi spiega come passare da una rappresentazione grafica ad una rappresentazione matriciale. Sarà su questa matrice che il metodo delle potenze sarà applicato.

Prima di passare alla descrizione del metodo riportiamo alcune definizione di necessaria importanza al fine di comprendere come il metodo delle potenze.

Definizione : Data una matrice quadrata A di ordine n, di elementi reali o complessi, si chiama auto valore di A un numero $\lambda \in \mathcal{K}$, per il quale esiste un vettore non nullo, $\boldsymbol{v} \in \mathcal{K}^n$, tale che

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$

Definizione : Une matrice A si dice non negativa se tutti i suoi elementi sono numeri reale non negativi, $\alpha_{i,j} \geq 0$

Il metodo delle potenze è particolarmente adatto per approssimare l'autovalore di modulo massimo, auto valore dominante, di una matrice e il corrispondente autovettore. Questo metodo trova un'applicazione in certi problemi specifici ed è alla base dell'algoritmo di Google di PageRank.

Supponiamo di avere una matrice quadrata A di ordine n. La matrice ha n auto valori e n autovettori associati agli auto valori. Gli auto vettori sono supposti linearmente indipendenti. Gli auto vettori, una volta calcolati sono:

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$$

λ_1 corrisponde all'autovalore dominante. Si può dimostrare che tutti gli tutti i rapporti $\frac{\lambda i}{\lambda 1}$ sono maggiorati da $\left|\frac{\lambda 2}{\lambda 1}\right|$ nel caso in cui $\alpha_1 \neq 0$, ossia $\mathbf{y_0}$ non ortogonale a $\mathbf{x_1}$. Gli x_i sono gli auto vettori associati agli auto valori e $\mathbf{y_0}$ è una combinazione lineate degli auto vettori:

$$\mathbf{y_0} = \sum \alpha_i \mathbf{x_i}\text{ , } i \in [1,n]$$

La velocità di convergenza del metodo dipende dai rapporti $\frac{\lambda i}{\lambda 1}$ e quindi dal rapporto $\left|\frac{\lambda 2}{\lambda 1}\right|$.

Algoritmi di Link Analysis Ranking

La mole d'informazione presente sulla rete Internet è cresciuta in maniera esponenziale negli ultimi decenni. I contenuti presenti sulla rete sono del tutto eterogenei, immagini, testi, video, metadati, e sono legati da un unico filone comune; i collegamenti ipertestuali fra loro. La crescita del numero d'informazione ha avuto come effetto primario la creazione di una barriera per gli utilizzatori nel trovare l'informazione di cui necessitano e ancora di più di avere un'insieme di risultati pertinenti. La branca dell'informatica che si occupa del recupero dell'informazione ha il nome di Information Retrival (IR). Si occupa non solamente della ricerca, ma anche della memorizzazione, rappresentazione e reperimento dei dati.

I motori di ricerca permettono essenzialmente di ritrovare le informazioni sul web. Il loro processo può essere schematizzato in tre passi:

I. Raccolta dei dati
II. Elaborazione dei dati raccolti
III. Fornire dei risultati a seguito di query da parte degli utilizzatori

E' nella seconda fase che ad ogni pagina, risorsa, un punteggio viene attribuito: ranking. Il valore del ranking specifica l'importanza della risorse permette l'indicizzazione dei dati raccolti.

Il ranking può essere visto come una funzione che associa un numero reale ad ogni pagina data una query.

Detto P l'insieme delle pagine e la Q la query fatta dall'utilizzatore:

$$r_q : P \rightarrow \mathcal{R}$$

Il primo passo per il calcolo del ranking è la trasformazione dell'insieme delle pagine P in un grafo orientato G = (V, E), dove ogni vertice corrisponde ad una pagina e ogni arco indica la presenza di un link tra due pagine. L'output dell'algoritmo sarà un vettore $\boldsymbol{\alpha}$= (α_1, …, α_n) dove n = |V| e ogni α_i con i ∈ (1,n) corrisponde al ranking associato alla i-esima pagina.

Primi algoritmi

Nel corso del tempo, diversi algoritmi sono stati creati per trovare una correlazione tra le risorse che sono sul web. Riportiamo due esempi di algoritmi che sono in un certo modo i precursori dell'algoritmo di PageRank; algoritmo Indegree, e algoritmo HITS.

Algoritmo Indegree

Questo algoritmo può essere considerato come il capostipite di tutti gli algoritmi di analisi dei link, considerando una pagina web secondo la sua popolarità. L'idea alla base dell'algoritmo è di attribuire un valore di popolarità che si basa unicamente sul numero di link entranti in una pagina e provenienti da altre pagine web. Più il numero di link entranti è alto e più una pagina sarà popolare. Per una generica pagina i definiamo la sua popolarità come:

$$PR(i) = |indegree(i)|$$

Algoritmo HITS

Questo algoritmo si differenzia dall'algoritmo precedente prendendo in considerazione i link entranti e uscenti creando due punteggi di popolarità per ogni pagina web.

Una pagina web è considerata una "hub page" se contiene molti link uscenti, si chiama, invece, "authority page" una pagina che ha molti link entranti. Una pagina può essere allo stesso tempo una "hub page" e una "authority page".

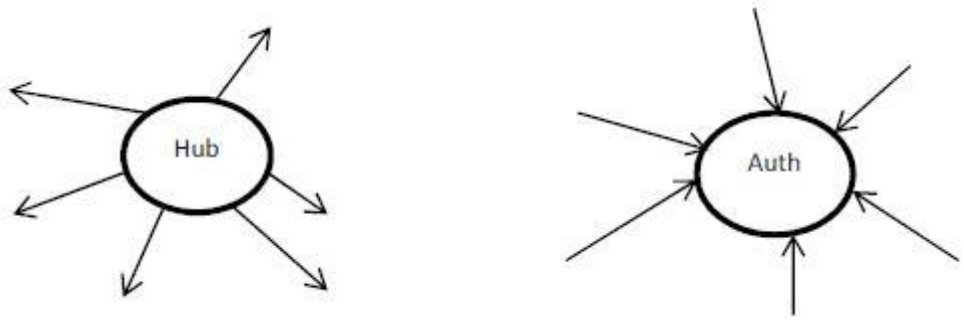

Figura 9 : hub e authority page

L'algoritmo assegna due punteggi ad ogni pagina:

- "hub score"
- "auth score"

I due punteggi sono complementari per il calcolo di popolarità della pagina. Definiamo come un buon "hub" una pagina che punta a buone "authority" e una buona "authority" una pagina puntata da buoni "hub".

L'algoritmo è di tipo iterativo. Allo stato iniziale sono assegnati due punteggi ad ogni pagina, le "hub score", y_i ,e le "auth score", x_i. Questi punteggi sono raffinati ad ogni iterazione. All'istante t le "auth score" ha come valore la somma di tutti gli "hub score" all'istante t-1 di tutte le pagine che puntano verso di lui. Le "hub score" all'istante t è dato dalla somma di tutti gli "auth score" allo stesso istante di tutte le pagina che sono puntate dalla nodo corrente. In termini matematici

$$x_i^{(k)} = \sum_{j:e_{j,i} \in E} y_j^{(k-1)}$$

$$y_i^{(k)} = \sum_{j:e_{i,j} \in E} x_j^{(k)}$$

Figura 10 : Hub score e Auth score

La forma delle equazione di calcolo si presta bene ad una rappresentazione matriciale. Il grafo è prima trasformato in matrice sfruttando la matrice di adiacenze e poi i calcoli degli score sono effettuati. La risoluzione del problema si trova ancora una volta nella ricerca degli autovettori della matrice. A questo scopo il metodo delle potenze viene applicato.

Detta P la matrice di adiacenza vogliamo trovare due auto vettori. Il primo deriva dalla matrice P^TP, che corrisponde al vettore di punteggi "authority", e il secondo dalla matrice PP^T, vettore dei punteggi "hub". Una volta ottenuti i due vettori li normalizziamo e li ordiamo in modo decrescente per ottenere il risultato finale del ranking.

Formula originale del Page Rank

Dato un insieme di pagine P definiamo il Page Rank di una generica pagina P_i , e la denotiamo con $r(P_i)$, è la somma dei Page Ranks di tutte le pagine che puntano a P_i:

$$r(P_i) = \sum_{Pj \in Bpi} \frac{r(P_j)}{|P_j|}$$

con B_{pi} l'insieme delle pagine che puntano verso P_i e con $|P_j|$ il numero di outlinks della pagina P_j. Il problema di questa equazione è il valore incognito dei $r(P_j)$. Per risolvere questo problema gli ideatore dell'algoritmo hanno assegnato un valore iniziale a $r(P_j)$, ossia $r_0(P_j)$= 1/n, dove n è il numero di pagine totali.

Una volta fissato il valore di ranking iniziale di ogni pagina possiamo calcolare iterativamente il nuovo valore di popolarità di ogni pagina mediante la formula che segue:

$$r_{k+1}(P_i) = \sum_{Pj \in Bpi} \frac{r_k(P_j)}{|P_j|}$$

L'algoritmo si ferma quando arriviamo a convergenza, ossia quando i valori di ogni $r_{k+1}(P_i)$ per ogni i non varia più. Per limitare il tempo di calcolo e per ridurre il numero di iterazioni la condizione che permette di stoppare l'algoritmo è meno restrittiva; l'algoritmo converge quando $| r_{k+1}(P_i) - r_k(P_i)| < \varepsilon \; \forall \; i \in E$ e ε un valore sufficientemente piccolo.

Il numero di iterazioni dipende dal numero di dati e dalla variazione del ranking di ogni pagina ad ogni iterazione. Il grafo che segue dà un'idea di come il numero di iterazione possa variare in funzione di questi due parametri.

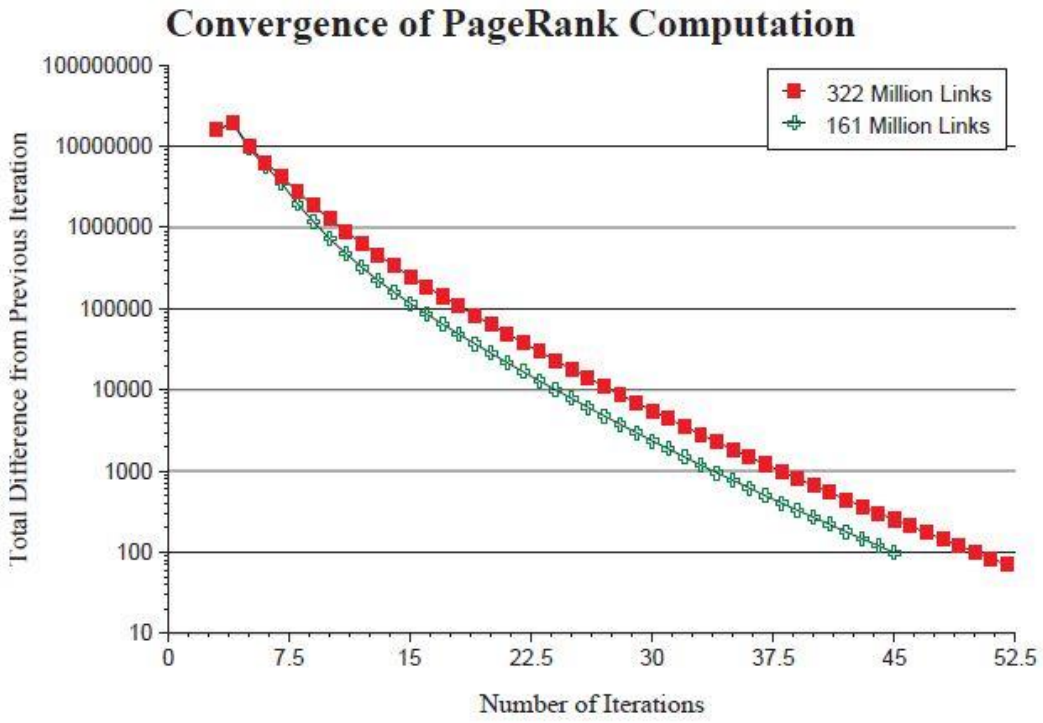

Figura 11 : Convergenza del Page Ranking

Algoritmo di Spanning Tree

L'algoritmo di Spannig Tree ha come scopo di trovare l'albero minimo a partire di un grafo orientato. Detto G = (V,E) il grafo orientato, l'albero minimo è dato dal sottoinsieme di archi $T \subseteq E$ che minimizzano la somma dei pesi sugli archi, c_e. Il calcolo dello spanning tree non può essere effettuato tramite un brute force, data la complessità del problema, n^{n-2}, teorema di Cayley.

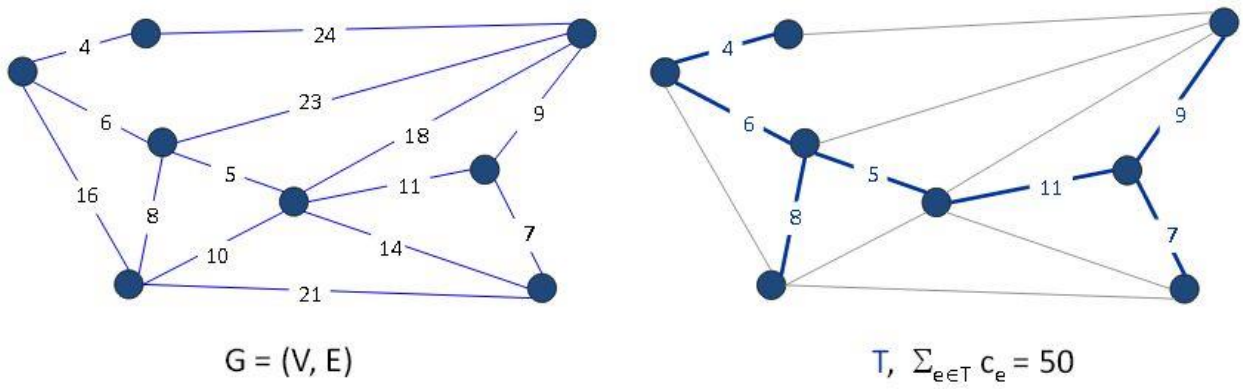

Figura 12 : Ricerca dello spanning tree su ungrafo.

La ricerca del minimo Spanning tree è un problema ricorrente che trova applicazione in diversi domini: design delle reti elettriche, di computer, di telefonia o ancora la risoluzione di problemi N-P difficili.

Il calcolo del MST (Minimun Spanning Tree) può essere fatto grazie a degli algoritmi detti “greedy”, golosi, come:

- Algoritmo di Prim
- Algoritmo di Kruscal

Dal punto di vista dell’implementazione verrà utilizzato l’algoritmo di Prim.

Algoritmo di Prim

L’algoritmo di Prim che permette il calcolo del MST è il seguente:

1. Scelta di un nodo radice e inizializzare l’insieme T a vuoto
2. Aggiungere un arco a T che è incidente solo su un nodo in T e con costo minimo

Esempio d’applicazione dell’algoritmo:

Stato Iniziale con T = {} e s come nodo radice

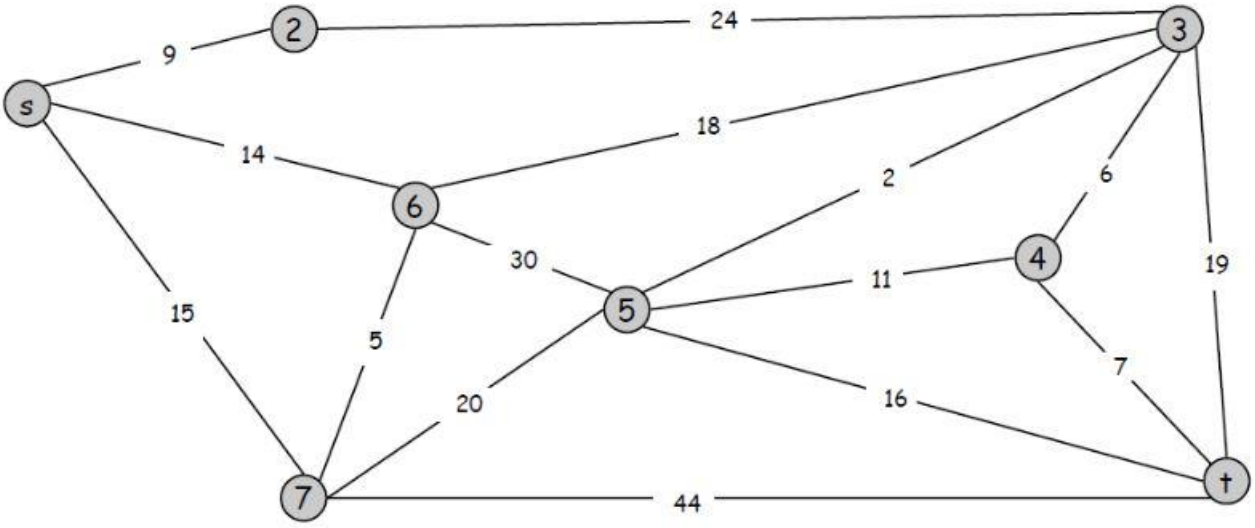

Figura 13 : Prim stato iniziale

Iterazione 1, T = {(s,2)} e costo totale = 9

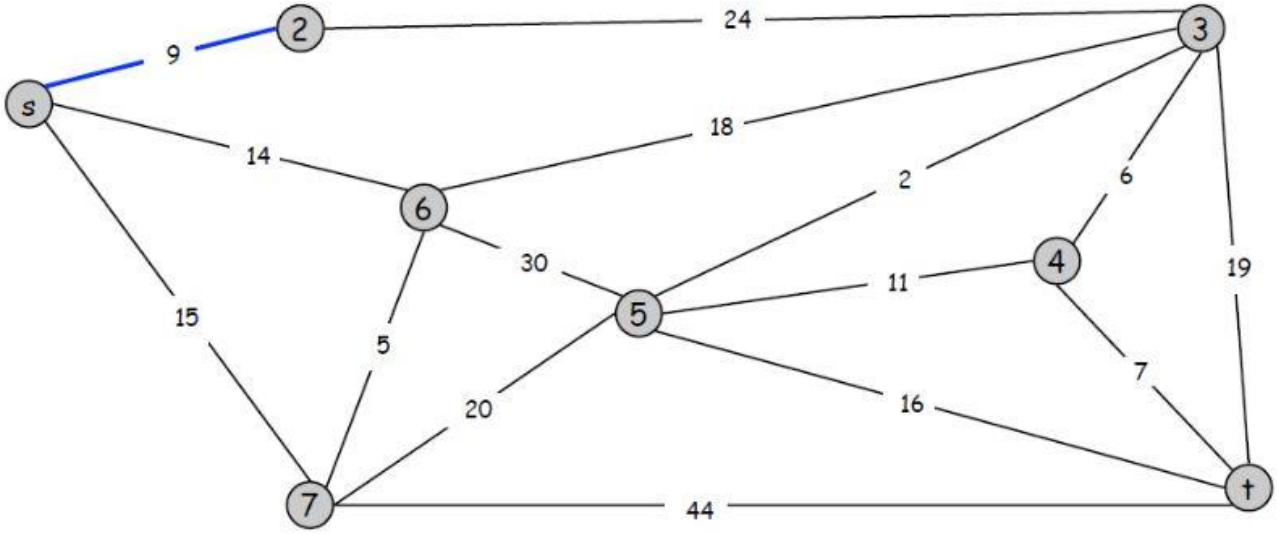

Figura 14 : Prim iterazione 1

Iterazione 2, T ={(s,2);(s,6)} e costo totale = 23

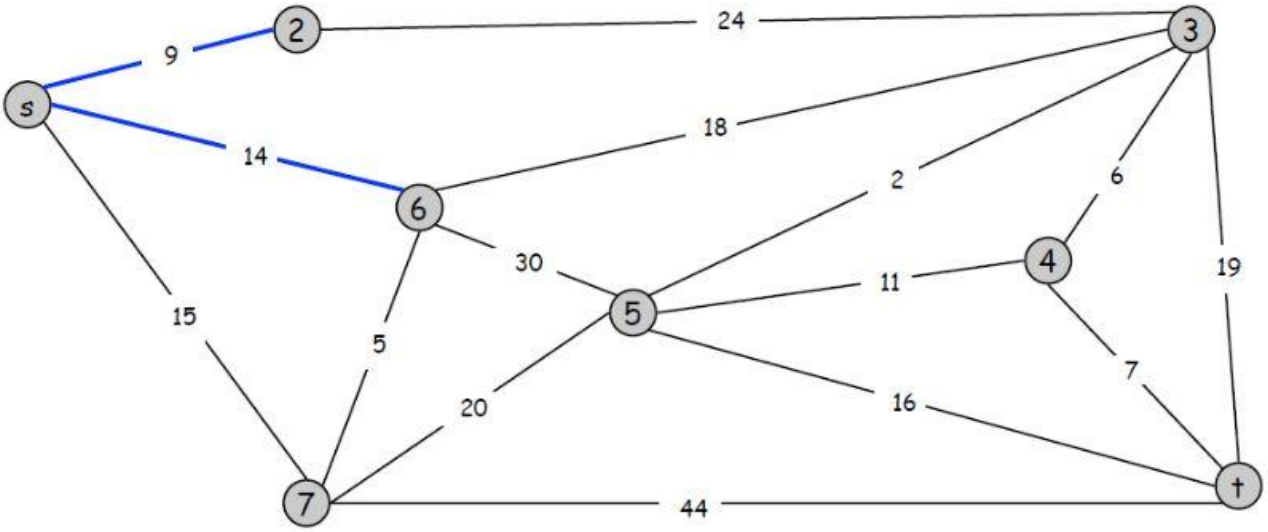

Figura 15 : Prim iterazione 2

Dopo 7 iterazioni il MST è ottenuto e ha come costo totale 61.

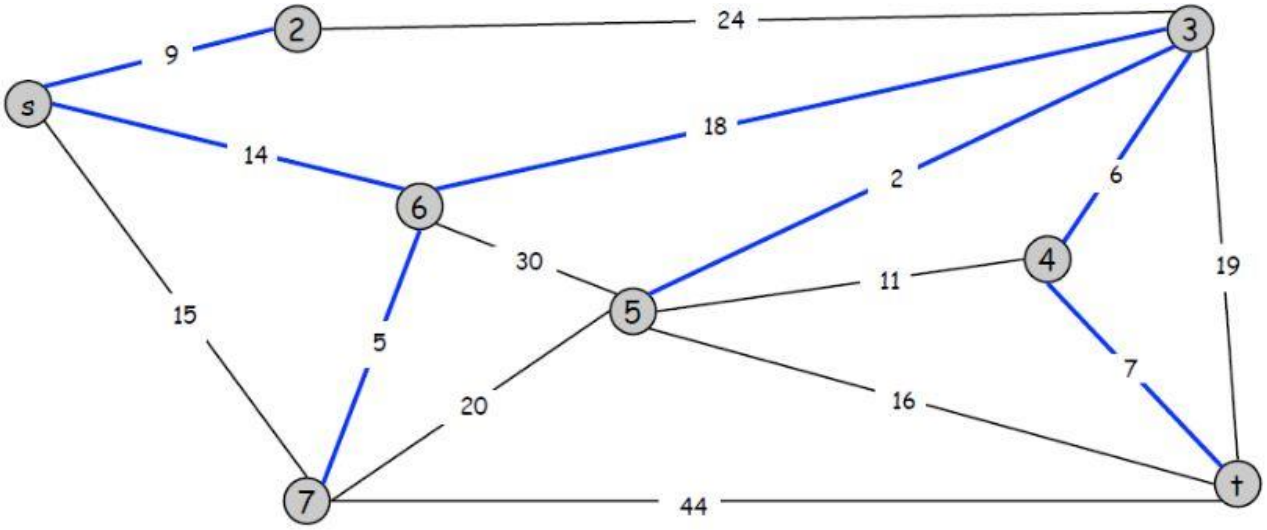

Figura 16 : MST ottenuto con l'algoritmo di Prim

Algoritmo di Kruscal

L'algoritmo di Kruscal che permette il calcolo del MST è il seguente:

1. Inizializzare l'insieme T a vuoto
2. Considerare gli archi in ordine di costo crescente
3. Aggiungere l'arco in T se non crea un ciclo

Esempio d'applicazione dell'algoritmo:

Il grafo allo stato iniziale è lo stesso utilizzato per l'algoritmo di Prim.

Iterazione 1, T = {(3,5)} e costo totale = 2

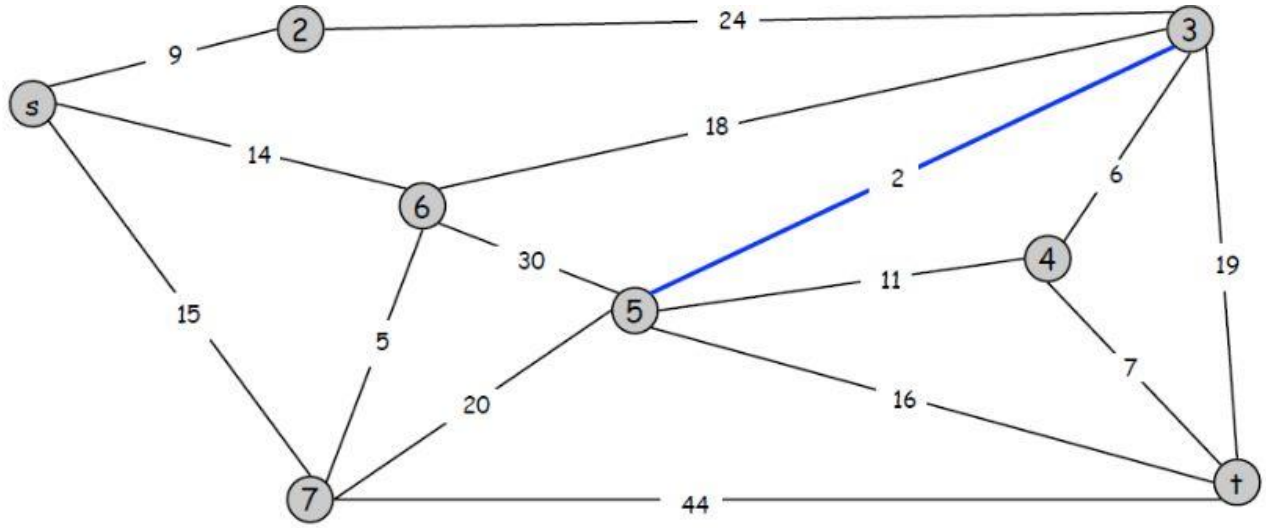

Figura 17 : Kruscal iterazione 1

Iterazione 2, T ={(3,5);(6,7)} e costo totale = 7

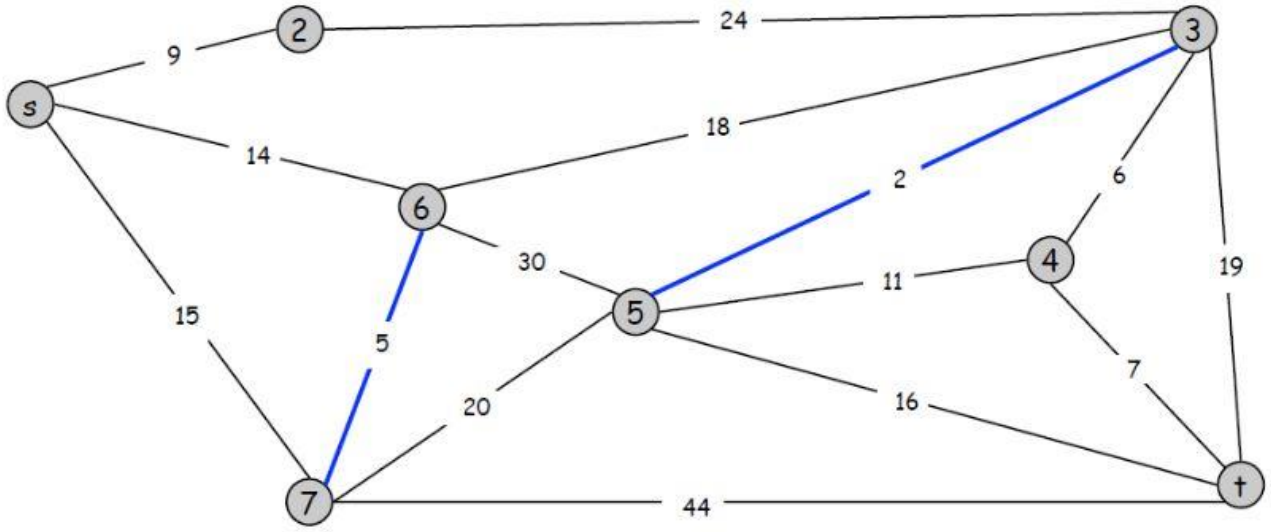

Figura 18 : Iterazione 2

Il MST ha come costo totale 61.

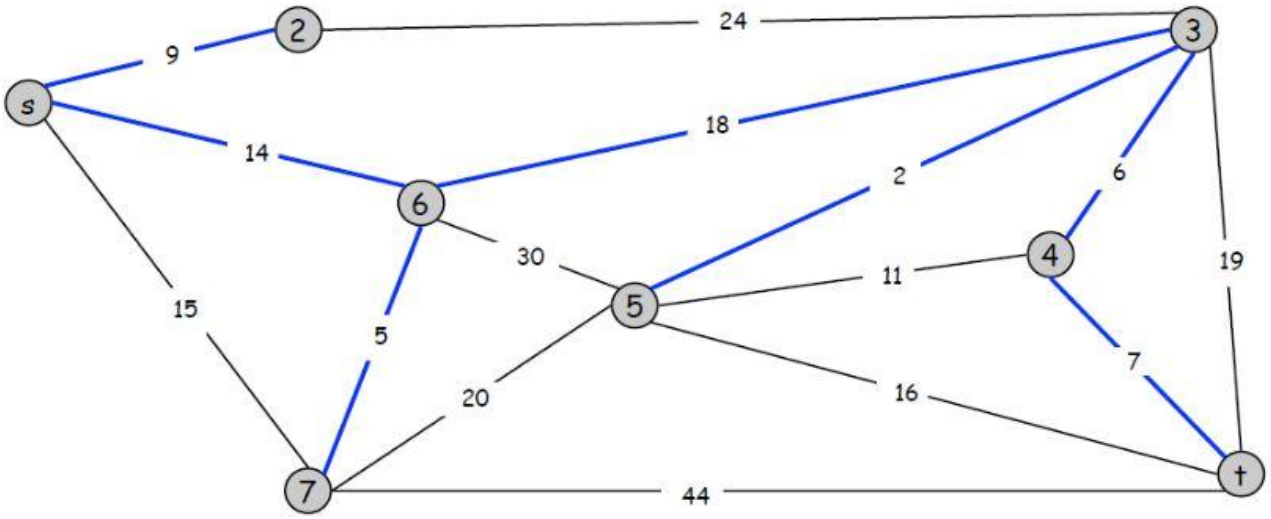

Figura 19 : MST ottenuto con l'algoritmo di Kruscal

Come giusto che sia i due algoritmi ritrovano lo stesso MST.

Applicazione al caso RDF

Questo lavoro apporta un contribuito al trattamento dei dati RDF, fornendo uno strumento che permette di determinare i collegamenti che esistono tra le diverse risorse. Inoltre, un altro aspetto importante di questo lavoro è l'ottimizzazione del percorso esistente tra le diverse risorse, tramite la realizzazione dello spanning tree.

Nel prossimo capitolo, presenteremo i differenti strumenti (RDF, XML) che sono saranno utili per la realizzazione dello strumento sofware.

Strumenti

Il modello dei dati RDF.

Il modello RDF e la specificazione della sintassi(diventata una raccomandazione della W3C dal 1999)definisce il modello di dati RDF e la sintassi di serializzazione basata su XML (6) (7). Uno degli elementi importanti di questo formato di dati è che permette anche di accedere a dati che hanno un formato diverso. Per fare ciò, in RDF si assegna ad ogni risorsa una URI(Uniform Resource Identifier), poi si scrivono degli statement che descrivono le proprietà della risorsa.

Questo modello di dati usa gli URI per descrivere in modo disambiguo gli aggetti, e le proprietà di tali oggetti permettono di sapere quali sono le relazioni tra quest'ultimi. Questo meccanismo semplice ed efficace permette quindi di rappresentare ed integrare informazioni, visto che permette di mettere in relazioni dati provenienti da sorgenti diverse.

Il cuore del modello RDF.

Il modello RDF è molto simile ai modelli di dati orientati oggetto (8). Questo modello è basato sulle entità, che sono rappresentate da URI come abbiamo visto nel paragrafo precedente, e di relazioni binarie o statement, che permettono di mettere in relazione le differenti entità.

Nella rappresentazione grafica di uno statement, la sorgente della relazione viene chiamata soggetto, l'arco etichettato è il predicato che può anche essere chiamato proprietà e la destinazione della relazione viene chiamato oggetto.

Gli statement e i predicati vengono chiamati "first-class object", ciò che significa che potrebbero essere usati come soggetti o oggetti di altri statement.

Il modello di dati RDF fa una chiara distinzione tra risorse (oggetti identificati da URI), ed i letterali (che sono principalmente delle stringhe).

Il soggetto ed il predicato di uno statement sono sempre risorse, mentre un oggetto potrebbe essere una risorsa o un letterale.

Nei diagrammi che descrivono i documenti RDF, le risorse vengono sempre disegnate come degli ovali mentre i letterali vengono rappresentati da scatole.

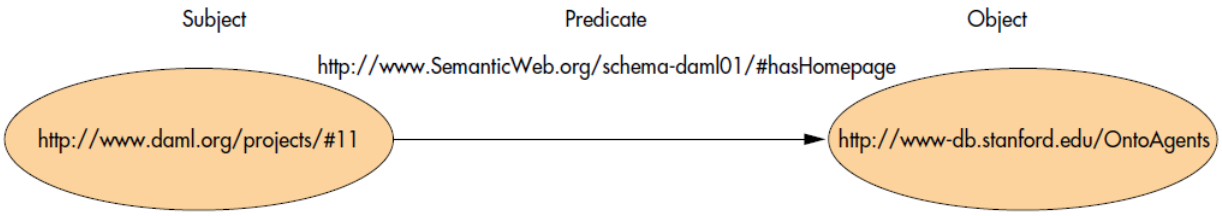

Figura 20 : Esempio di statement RDF

La figura precedente illustra la relazione esistente tra un soggetto (http://www.daml.org/projects/#11) ,ed un oggetto(http://www-db.stanford.edu/OntoAgents). Il predicato che descrive la relazione tra le due entità è indicato sull'arco.

Il fatto che i predicati siano rappresentati da URI aiuta a rimuovere tutte le ambiguità possibili. Questo è dovuto al fatto che l'uso di parole chiave non garantisce sempre la flessibilità, perché quest'ultime possono variare nel tempo. L'identificazione dei predicati con URI aiuta quindi a fare evolvere l'uso del formato RDF, perché un cambiamento di un predicato significherebbe semplicemente di cambiare l'URI che rappresenta tale predicato.

Vocabolario additivo.

La sintassi XML-namespace viene usata per abbreviare gli URI che sono usati negli statement.Nell'esempio precedente, si può sostituire una parte dell'URI del predicato con un prefisso. Ad esempio, la stringa seguente: http://www.SemanticWeb.org/schema-daml01/#hasHomepage può essere sostituita con il simbolo sw.

L'URI del predicato sarà quindi: sw:hasHomepage. Come si può vedere, questa notazione permette di semplificare notevolmente la scrittura di un documento RDF.

Il namespace RDF definisce la proprietà "rdf:type". Questa proprietà permette di sapere quali sono le relazioni tra le differenti risorse del documento RDF. Questa proprietà è particolarmente utile

quando viene usata con le parole chiave “Class” e “subClassOf” del vocabolario RDF. La proprietà “rdf:type” è simile alla proprietà “instanceOf” dei linguaggi orientati oggetto.

Un altro aspetto importante del formato RDF è la rappresentazione delle collezioni. In effetti, le collezioni possono esser viste come degli insiemi. Ad esempio i dipendenti di un dipartimento in un’azienda possono esser visti come membri di una collezione, e quest’ultimi potrebbero avere gli stessi diritti. C’è quindi la possibilità di costruire uno statement per ogni dipendente, oppure di fare uno statement unico per tutti i dipendenti del dipartimento usando un “container”.

In RDF, ci sono tre tipi di “container”. Questi container possono rappresentare risorse o letterali. Di seguito, verranno descritti questi “container”:

- Bags: sono delle liste non ordinate.
- Sequences: sono delle liste ordinate. Ad esempio, una lista di lavori a realizzare in un processo può essere memorizzata con questo tipo di elemento.
- Alternatives: sono liste che permettono di impostare il valore di una proprietà.

Per riferirsi ad uno statement in un documento RDF, è necessario trattarlo come una risorsa. L’associazione di uno statement e di una specifica risorsa che rappresenta questo statement viene chiamata “reification”.

Lo statement che è stato reificato viene chiamato “reified statement”.

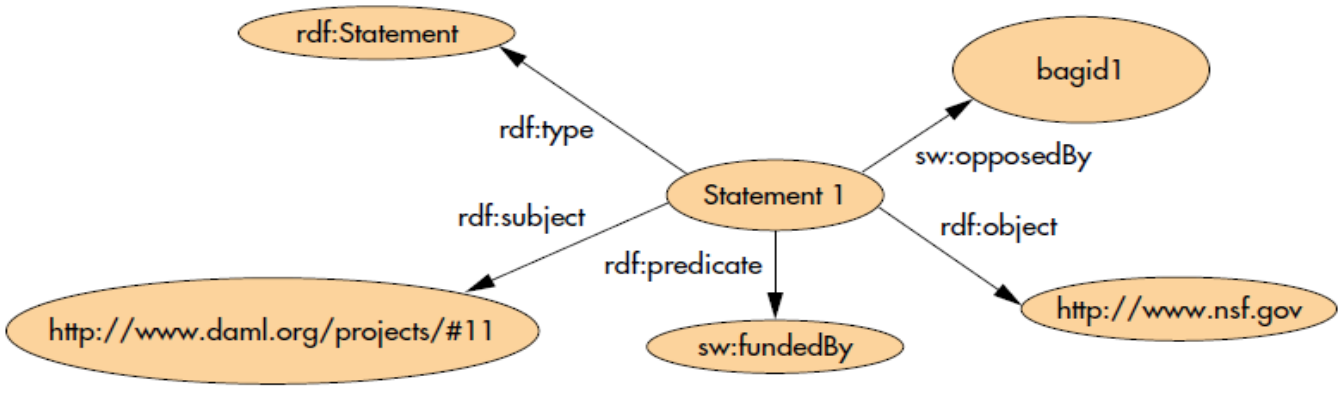

Figura 21 : Esempio di utilizzo di uno statement reificato.

Schema RDF

Si tratta di un'applicazione del formatto RDF che introduce il concetto orientato oggetto al formatto RDF. Lo schema RDF fornisce i mezzi per definire la proprietà di domini, intervalli, classi e sottoclassi (Class, SubClass).

In uno schema RDF, le proprietà non sono locali. Bensì, sono globali e definite in termini di classi che sono connesse tra di loro.

Tutte le proprietà sono definite con il tag "rdfs:property". Bisogna notare che il dominio e l'intervallo di una proprietà possono subire restrizioni usando le proprietà "rdfs-domain" e "rdfs:range".

La specifica RDF definisce altre primitive del linguaggio:

- rdfs:label: permette di definire un formatto di nome comprensibile dall'utilizzatore.
- rdfs:comments: permette agli sviluppatori di fare dei commenti.
- rdfs: subPropertyOf: questa proprietà indica che la proprietà viene inglobata in un'altra proprietà. Ad esempio, la proprietà "fatherOf" può essere inglobata nella proprietà "parentOf".
- rdfs: seeAlso e rdfs: isDefinedBy: indicano le pagine web collegate alla pagina presente che contengono informazione additiva
- rdfs: ConstraintResource e rdfs:ConstraintProperty: indicano vincoli sofisticati che non sono rispettati dallo schema RDF.

Il ruolo dello schema RDF è di definire un vocabolario associato ad un particolare namespace all'URI del namespace. Lo schema RDF permette quindi agli sviluppatori di definire un vocabolario che dovranno rispettare tutti i documenti RDF.

Sintassi RDF

Lo scambio di dati che rappresenta il formatto RDF dovrebbe essere facilitato tramite la sintassi. Per agevolare lo scambio di dati, la scelta che è stata fatta è di utilizzare l'XML.

Tuttavia, il modello dei dati RDF non è legato a nessuna sintassi particolare, quindi può essere espresso in qualsiasi rappresentazione sintattica. Questo modello presenta anche il vantaggio che può essere realizzato a partire da sorgenti di dati nonRDF.

La sintassi XML di serializzazione di documenti RDF è difficile da capire, ma le APIs (Application Programming Interfaces) di RDF permettono agli sviluppatori di non interessarsi ai dettagli della sintassi di serializzazione e di trattare i dati RDF come dei grafi.

La specifica suggerisce due sintassi per la serializzazione di dati RDF in XML: abbreviata e standard. Entrambi le sintassi usano meccanismi di namespace XML per abbreviare gli URI. Si può quindi intuire che la sintassi XML assomiglia ai documenti XML.

A questo punto, risulta chiaro che il formatto RDF è il minimo comune denominatore per realizzare l'interoperazione tra le applicazioni web. Essendo un modello orientato oggetto, fornisce un modello più conveniente dell'XML per lo scambio di informazioni, ed è più flessibile per la definizione di nuovi vocabolari. RDF risulta quindi essere uno strumento molto importante per la prossima fase di sviluppo del web.

Nei paragrafi precedenti, abbiamo presentato il modello RDF, ed abbiamo mostrato la sua importanza. Nel prossimo paragrafo, presenteremo in modo breve l'obiettivo di questo lavoro.

Il formato XML

Definizione

Il formato XML (Extensible Markup Language) è una tecnologia usata per le applicazioni WEB. Si tratta di uno standard W3C (World Wide Web Consortium), che permette ad ogni utilizzatore di creare i propri tag (etichette).

XML permette quindi di semplificare le transazioni tra le differenti entità presenti sul web. Il primo obiettivo di questo standard è facilitare lo scambio automatizzato di contenuti complessi (alberi, documenti ricchi di contenuto…) tra sistemi di informazioni eterogenei.

Questo ci sarà utile per memorizzare il contenuto del risultato(albero risultato dell'algoritmo dello spanning tree) che otterremo alla fine del trattamento dei file RDF.

Sintassi di un documento XML

In questo paragrafo, spiegheremo com'è fatto un documento XML. Come detto precedentemente, è il formato che è stato scelto per il salvataggio del risultato del trattamento che verrà fatto sui documenti RDF.

La figura seguente rappresenta un esempio molto semplice di documento XML. Questo esempio ci sarà utile per la comprensione della struttura dei documenti XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<lavoratori>
    <lavoratore>
        <nome>Luca</nome>
        <cognome>Cicci</cognome>
        <indirizzo>Milano</indirizzo>
        <paese>Italia</paese>
    </lavoratore>
    <lavoratore>
        <nome>Max</nome>
        <cognome>Rossi</cognome>
        <indirizzo>Roma</indirizzo>
        <paese>Italia</paese>
    </lavoratore>
    <lavoratore>
        <nome>Louis</nome>
        <cognome>Blanc</cognome>
        <indirizzo>Parigi</indirizzo>
        <paese>Francia</paese>
    </lavoratore>
</lavoratori>
```

Figura 22 : Esempio di documento XML.

La prima riga di questo esempio indica la versione di XML in uso e specifica la codifica UTF-8 per la corretta interpretazione dei dati.

In XML, alcuni caratteri speciali hanno una rappresentazione specifica. La tabella seguente ci fa vere la rappresentazione di alcuni dei caratteri speciali.

Carattere	Entità
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

Figura 23 : Caratteri speciali in XML.

Dopo aver visto la struttura di un documento XML e la rappresentazione di alcuni caratteri speciali, ci interessiamo adesso ai diversi elementi che compongono il documento.

- **I tag o le etichette**: L'XML, come l'HTML, utilizza dei marcatori, che vengono anche chiamati *tag* (etichette), per assegnare una semantica al testo. I tag possono contenere informazioni in due modi: attraverso dei parametri oppure racchiudendo del testo o altri tipi di informazioni. Essi tendono a costituire quello che in gergo viene chiamato un linguaggio ben parentesizzato (altrimenti il documento non è da considerarsi ben formato). Segue che possono essere tag di apertura, necessariamente seguiti da tag di chiusura (tra i quali si può avere un contenuto) oppure tag che si aprono e chiudono, e possono quindi fornire informazioni solo attraverso i loro parametri.
  Ogni etichetta inizia e finisce con delle parentesi angolari <> (che in altri contesti sarebbero i segni di minore e maggiore), mentre la chiusura del tag o il tag di chiusura è rappresentato dalla barra /.

- Per essere correttamente interpretato, un documento XML deve esser ben formato, quindi deve possedere le seguenti caratteristiche:
  - Un **prologo**, che è la prima istruzione che appare scritta nel documento. Nel nostro caso: <?xml version="1.0" encoding="UTF-8"?>.
  - Un unico **elemento radice** (ovvero il nodo principale, chiamato *root element*) che contiene tutti gli altri nodi del documento. Nel nostro esempio: <utenti>.

- All'interno del documento tutti i **tag** devono essere bilanciati.

Tecnologie in rapporto con il formato XML

Ci sono diverse tecnologie che permettono di scrivere dei documenti corretti in formato XML. Le tecnologie più usate sono le seguenti:

- **DTD:** (acronimo di Document Type Definition) è un documento che permette di specificare le caratteristiche strutturali di un documento scritto in formato XML, attraverso una serie di "regole grammaticali". In particolare definisce l'insieme degli elementi del documento XML, le relazioni gerarchiche tra gli elementi, l'ordine di apparizione nel documento XML e quali elementi e quali attributi sono opzionali o meno.
  Il seguente DTD:

```
<! ELEMENT azienda (nomeAzienda, partitaIVA)>
<! ELEMENT nomeAzienda (#PCDATA) >
<! ELEMENT partitaIVA (#PCDATA) >
```

  definisce una struttura XML così composta:

```
<azienda>
<nomeAzienda>Politecnico di Milano</nomeAzienda>
<partitaIVA>25487522651</partitaIVA>
</azienda>
```

  Tuttavia, è importante notare che il DTD è opzionale, quindi ci possono essere dei documenti XML che non hanno un DTD associato.

- **XML Schema**: è simile alla DTD, viene quindi usata per definire la struttura di un documento XML. Oggi il W3C consiglia di adottarlo al posto della DTD stessa, essendo una tecnica più recente ed avanzata. La sua sigla è XSD, acronimo di XML Schema Definition.

- **XLink**: serve a collegare in modo completo due documenti XML; al contrario dei classici collegamenti ipertestuali che conosciamo in HTML, XLink permette di creare link multi direzionali e semanticamente avanzati.
- **XLS**: (acronimo di eXtensible Stylesheet Language) è il linguaggio con cui si descrive il foglio di stile di un documento XML. La sua versione estesa è l'XSLT (dove la T sta per Transformations).
- **Xpath**: è un linguaggio con cui è possibile individuare porzioni di un documento XML e sta alla base di altri strumenti per l'XML come XQuery.
  A supporto di questo scopo principale, fornisce anche elementari funzionalità per trattare stringhe, numeri e dati booleani. Il suo funzionamento si basa sulla creazione di un albero a partire dal documento e la sintassi succinta permette di indirizzare una specifica parte attraverso i nodi dell'albero con la semplice parola path.
- **Xpointer**: serve ad identificare univocamente precise porzioni di un documento XML; consente poi il loro accesso ad altri linguaggi o oggetti di interfaccia.
- **DOM**: è un'interfaccia di programmazione, come SAX, implementata in una moltitudine di linguaggi di programmazione, per la manipolazione di file XML. DOM costruisce partendo dal file XML un albero dove ogni nodo dell'albero corrisponde ad un elemento del file; per questo motivo è detta tree based.

Dopo aver presentato l'algoritmo di spanning tree e i differenti formati di dati, nel capitolo successivo, presenteremo il lavoro che abbiamo realizzato. In effetti, il nostro lavoro si basa sui concetti appena definiti.

Il nostro lavoro

Introduzione

L'obiettivo di questo lavoro è di fornire un software che permette di trasformare i documenti RDF in grafi non orientati, di assegnare dei pesi a tutti gli archi del grafo ottenuto a partire da ogni documento. Successivamente, viene applicato l'algoritmo di spanning tree sul grafo ottenuto, con lo scopo di trovare il percorso ottimale nel percorrere le differenti risorse che costituiscono il grafo. L'ultima tappa del lavoro consiste nel memorizzare il risultato dell'algoritmo di spanning tree in un file XML.

In questo capitolo, descriveremo il procedimento che è stato utilizzato per raggiungere gli obiettivi fissati.

Descrizione del lavoro realizzato

In questa sessione, presenteremo le parti importanti del lavoro che abbiamo effettuato. La figura seguente riassume le grandi linee del lavoro che è stato fatto per poter ottenere il risultato desiderato:

Figura 24 : Rappresentazione grafica del lavoro realizzato.

In seguito, descriveremo ognuna delle parti che sono state realizzate.

Parsing e filtraggio del documento RDF.

In informatica, il parsing o analisi sintattica è il processo atto ad analizzare uno stream continuo in input (letto per esempio da un file o una tastiera) in modo da determinare la sua struttura

grammaticale grazie ad una data grammatica formale. Un parser è un programma che esegue questo compito. Nel nostro caso lo stream di dati in ingresso è rappresentato da un file RDF. La codifica di questo file è stata specificata nel capitolo precedente. Lo scopo di questo primo passaggio è di riportare in una forma utilizzabile i dati contenuti nel file RDF. Questo primo procedimento è essenziale ed è presente in quasi tutti i processi di analisi dei dati, indipendentemente dalla loro forma d'origine e sorgente.

Dal punto di vista dell'implementazione il parsing è stato realizzato grazie alla libreria JENA(9). Questa libreria è stata implementata da Apache ed è ad uso libero.

Le principali API messe a disposizione dalla libreria sono il caricamento in memoria di un file RDF, la sua manipolazione e la riscrittura come file RDF o come XML. La classe che permette di rappresentare il modello RDF in Java è ModelFactory. L'utilizzatore, o meglio lo sviluppatore, può creare un'istanza vuota di un modello e riempirla con dei dati oppure creare un modello a partire da uno stream, ad esempio un file di testo. Nel nostro caso il modello è caricato tramite un file di testo che l'utilizzatore scegli grazie all'interfaccia grafica che gli è messa a disposizione.

Il modello creato da Jena rispecchia la forma standard di rappresentazione dei file RDF, ovvero ogni Statement, oggetti che fanno parte del modello, rappresenta una tripla, Soggetto, Predicato e Oggetto. Per ogni statement lo sviluppatore può avere accesso a diverse informazione che riguardano una delle tre componenti; lo sviluppatore può ottenere tutti gli Statement Incidenti o Puntati dallo Statement corrente e può scegliere diversi modi di rappresentare alla console la tripla di dati. Il recupero dei dati presenti nel modello può essere fatto in diversi modi: in maniera esaustiva tramite un iteratore sul modello che continuerà a ritornare uno Statement o tramite una "query". Si può ricorrere a diverse funzioni che prendono in entrata un Predicato, un Soggetto o un Oggetto e recuperare gli Statement associati.

Filtraggio di alcuni elementi

Dopo aver caricato il documento in memoria grazie al parsing e aver creato il modello corrispondente al file RDF dato in ingresso, l'operazione successiva consiste ad eliminare i dati che non utili all'analisi. Nel nostro caso i dati non rilevanti sono dei predicati che graficamente corrispondono agli archi che collegano i nodi del nostro grafo, soggetti o oggetti che siano.L'utilizzatore ha due scelte per determinare quali predicati escludere dall'analisi: scegliere

l'opzione per default, che corrisponde ad un insieme composto da tutti le parole chiavi del linguaggio RDF oppure creare un file con una lista con tutti i predicati che non vuole conservare nel corso del processo di analisi. Abbiamo dato all'utilizzatore la possibilità di avere una modalità ibrida; eliminare tutti i predicati che hanno un match nel dizionario di default e in più di aggiungere a questa lista la sua lista personalizzata. La lista dell'utilizzatore corrisponde ad un file di testo caricato tramite l'interfaccia grafica prima. Il file deve contenere una parole chiave per linea.

L'operazione di filtraggio è implementata tramite un'iterazione su tutti gli Statement. Se il predicato dello Statement contiene un Predicato presente nella lista delle parole chiavi da eliminare allora lo Statement sarà eliminato dal modello.

Dal punto di vista computazionale l'operazione di filtraggio cosi implementata ha una complessità pari a*O(n*m)*, dove n è la cardinalità degli Statement che compongono il modello e m è la cardinalità dell'insieme delle parole chiavi che non vogliamo conservare.

Anche se onerosa, dal punto di vista computazionale, questa operazione è fatta una sola volta ed è necessaria per ottenere in uscita dei dati che sono interessanti per l'analisi dell'utilizzatori.

Conversione in grafo

I documenti RDF hanno tre rappresentazioni:

1. File standard RDF
2. Rappresentazione in triple
3. Rappresentazione mediante un grafo orientato

Nel nostro caso solo due di queste rappresentazioni saranno utilizzate, file RDF per il parsing e la rappresentazione tramite un grafo orientato. La libreria Jena non permette di visualizzare in forma di grafo il modello creato. A fine di poter ottenere un grafo diverse manipolazioni sono state effettuate.

Per facilitare l'applicazione successiva d'all'algoritmo di Page Rank il modello è stato trasformato in matrice, e più precisamente in una matrice d'adiacenza. Ricordiamo dal capitolo sulla teoria dei

grafi che la rappresentazione di un grafo tramite la matrice d'adiacenza o tramite nodi e archi è completamente equivalente.

Durante il processo di trasformazione un dizionario è stato creato per associare ad ogni componete di uno Statement un identificativo in modo da poter ritrovare successivamente l'informazione testuale associata ad ogni Soggetto, Oggetto e Predicato. Questa operazione è fatta essenzialmente per due motivi; ritrovare più velocemente il valore testuale di un nodo o di un arco e ritrovare l'etichetta da associare ad un arco. Come specificato nei requisiti della tesina ogni arco ha due valori. Il primo è numerico è corrisponde al peso associato all'arco e il secondo testuale è il nome originale dell'arco, ovvero il predicato.

L'ultimo passaggio necessario per utilizzare la libreria JUNG (10) che ci permetterà di rappresentare in maniera grafica il grafo è la creazione di due classi; la classe Node (Nodo) e la classe Edge (Arco).

Nel nostro modello un nodo corrisponde ad un Soggetto o ad un Oggetto, e creiamo un arco per ogni collegamento esistente tra un Soggetto e un altro Soggetto o tra un Soggetto e un Oggetto. Dal punto di vista matriciale vuol dire creare un arco per ogni coppia di valori i,j della matrice con valore diverso da 0. La rappresentazione in forma matriciale dei dati in Java è stato ottenuto tramite la classe Matrix della libreria JAMA (11).

La libreria JUNG fornisce diversi tipi di modelli di grafi. La classe da noi utilizzata sarà UndirectedSparseGraph<?,?>. Tutti i modelli forniti sono dal punto di vista della programmazione Java generics, ovvero accettano qualsiasi tipo di classe a patto che implementino i seguenti metodi:

- Per una classe di tipo Nodo, primo parametro (?)
    - equals
    - comapreTo
    - toString
- Per una classe di tipo Arco, secondo parametro
    - equals
    - toString

Oltre al tipo di grafo la libreria offre una vasta scelta di tipi di layout per la rappresentazione grafica.

Calcolo dei pesi degli archi del grafo

Al fine di poter applicare l'algoritmo di Spanning Tree per l'albero di supporto bisogna attribuire ad ogni arco un peso. L'unica condizione imposta per il calcolo dei pesi sugli archi è che gli archi che hanno delle parole provenienti dal dizionario RDF devono avere un valore più alto.

I passi che sono stati fatti al fine di ottenere un grafo non orientato con dei pesi sugli archi sono i seguenti:

1. Utilizzazione dell'algoritmo di Page Rank per ottenere un valore di popolarità di ogni nodo
2. Calcolo dei pesi di ogni arco basandosi sul valore di popolarità dei nodi
    a. Penalizzazione nel caso l'arco abbia una parola appartenente al dizionario RDF
3. Trasformazione della matrice di adiacenza in una matrice diagonale superiore
4. Calcolo dell'inverso di ogni valore diverso da zero della matrice

Il punto numero uno, in cui l'algoritmo di Page Rank viene impiegato, ha come risultato una matrice dove la diagonale è nulla e i valori di una generica linea i indicano il valore di ranking del nodo i. Per ogni linea i il valore (i,j) è pari a

$$\begin{cases} 0 \; se \; i = j \\ 0 \; se \; non \; esiste \; la \; connessione \; dal \; nodo \; i \; al \; nodo \; j \\ r(P_i) \; se \; esiste \; la \; connessione \; dal \; nodo \; i \; al \; nodo \; j \end{cases}$$

L'algoritmo di Page Rank non prende in considerazione i link che rimandano alla stessa pagina o allo stesso sito. Nel nostro caso l'unica condizione che è stata tenuta è di non considerare i link che rimandano sulla stessa risorsa, auto cicli sul grafo.

La matrice cosi costruita non è ancora diagonale superiore né prende in conto le parole proveniente dal dizionario RDF. La penalizzazione é data dalla diminuzione del valore dell'arco. Il valore della penalizzazione è pari a un quarto della media delle pagine. Il passo successivo è rendere la matrice diagonale superiore e di rendere il grafo non orientato. Dal punto di vista grafico vuol dire fondere i due archi orientati, nel caso esistano, in un unico arco non orientato che colleghi il nodo i al nodo j. L'arco risultante avrà come peso la somma dei pesi dei due archi. In termini matematici otteniamo la seguente trasformazione:

Data la matrice quadrata M di dimensione n ∀ (i,j) con i,j ∈ [1,n] e i ≠ j

$$\begin{cases} (i,j) = (i,j) + (j,i) \; se \; i > j \\ 0 \; altrimenti \end{cases}$$

La matrice cosi costruita non è più interpretata come una matrice di adiacenza. L'esistenza di una calore (i,j) > 0 indica la presenza di un arco non orientato che collega il nodo i al nodo j.

A seguito dell'applicazione di Page Rank e dei passaggi su riportati il risultato che si ottiene é un grafo dove il peso di ogni arco indica la sua importanza, ovvero più il valore del peso è alto più esso è importate. L'algoritmo di Spanning Tree cerca invece gli archi di costo più basso per creare l'albero di supporto. Per poter applicare l'algoritmo in maniera che il risultato ottenuto sia quello desiderato, ovvero un grafo di costo minimo dove gli archi con popolarità più alti siano preservati abbiamo deciso che il peso di ogni arco sia uguale al suo reciproco. Una volta calcolato il reciproco possiamo applicare l'algoritmo di Spanning Tree e ottenere il risultato finale.

Applicazione dell'algoritmo di Spanning Tree

Questa parte rappresenta un elemento molto importante del nostro lavoro. Come detto all'inizio di questo rapporto, l'obiettivo ricercato era di trovare un collegamento ottimale tra le differenti risorse. L'algoritmo di spanning tree ci ha permesso di realizzare tale compito una volta determinati i pesi per ogni arco.

Il collegamento ottimale per collegare le risorse è dato dall'albero di supporto a costo minimo trovato a partire dal grafo a cui è stato applicato l'algoritmo di Page Rank e il calcolo dei pesi degli archi tramite la popolarità di ogni risorsa e la funzione di penalizzazione per gli archi che hanno una parola chiave appartenente al dizionario RDF.

L'implementazione di questa parte in Java corrisponde al passaggio dalla classe Graph alla classe Forest, che rappresenta l'albero. Come spiegato nel capitolo sulla teoria dei grafi esistono diversi modi, algoritmi per calcolare il MST. Nel nostro caso la funzione utilizzata è MinimumSpanningForest che prende in ingresso la classe Graph e che dà in uscita una classe Forest. Dal punto di vista dell'implementazione interna della funzione l'algoritmo che viene applicato per la ricerca dell'albero è l'algoritmo di Prim.

Di seguito riportiamo un esempio di grafo sul quale è stato applicato l'algoritmo di Spanning Tree.

- **Descrizione del data :** si tratta di una pagina web che contiene i link alle opere del poeta Dante Gabriel Rossetti. Questo documento è stato preso da un sito che contiene degli esempi di file RDF (12).

- **Numero di nodi:** 68 nodi prima del filtraggio.
- **Numero di statement:** 165 statement prima del filtraggio.

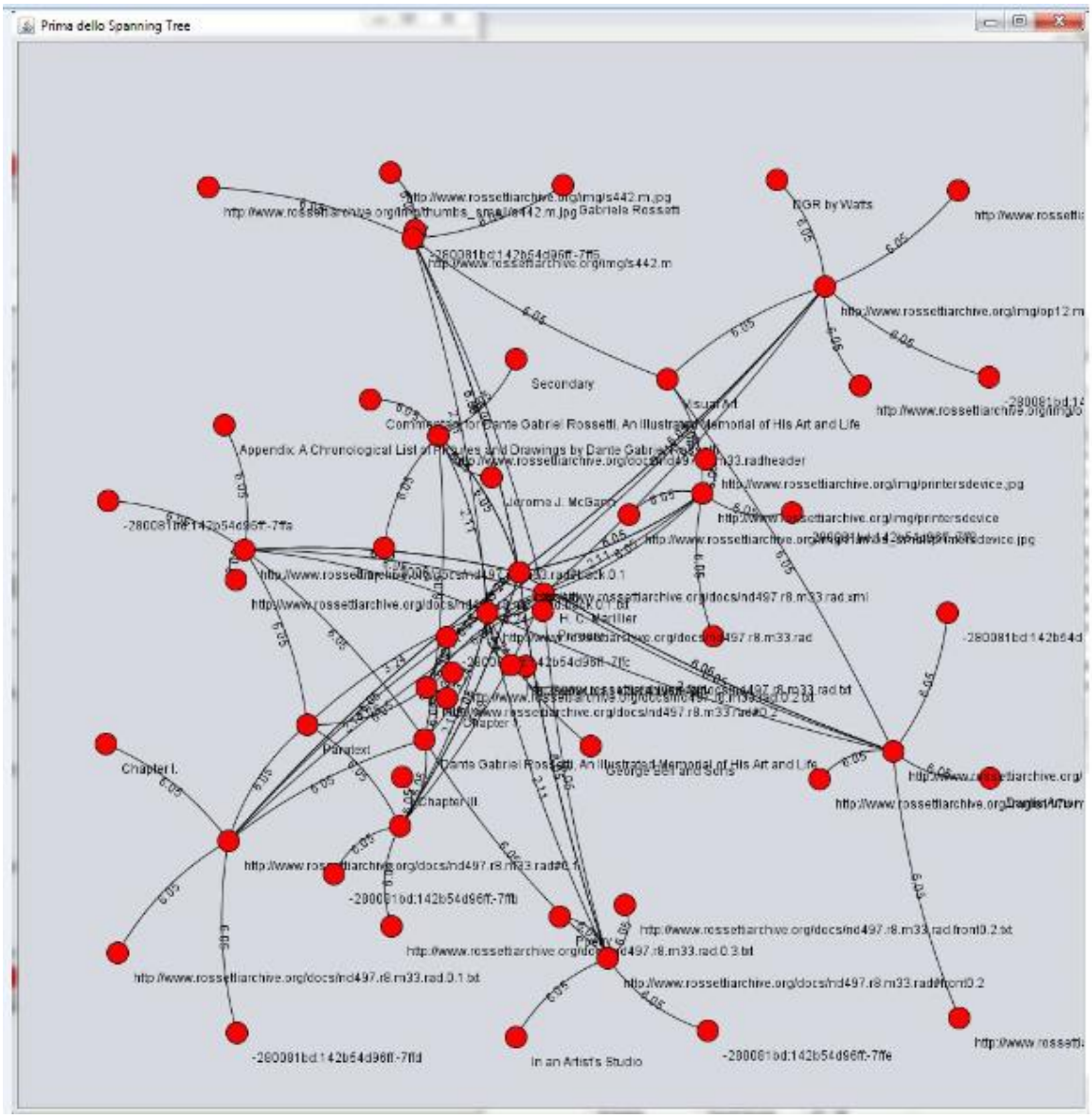

Figura 25 : Grafo ottenuto prima dello spanning tree.

La matrice di adiacenza è stata utilizzata come input per poter applicare l'algoritmo del page ranking. Dopo l'algoritmo del page ranking, il grafo è stato trasformato da orientato a non orientato, ciò che dà come risultato la figura precedente.

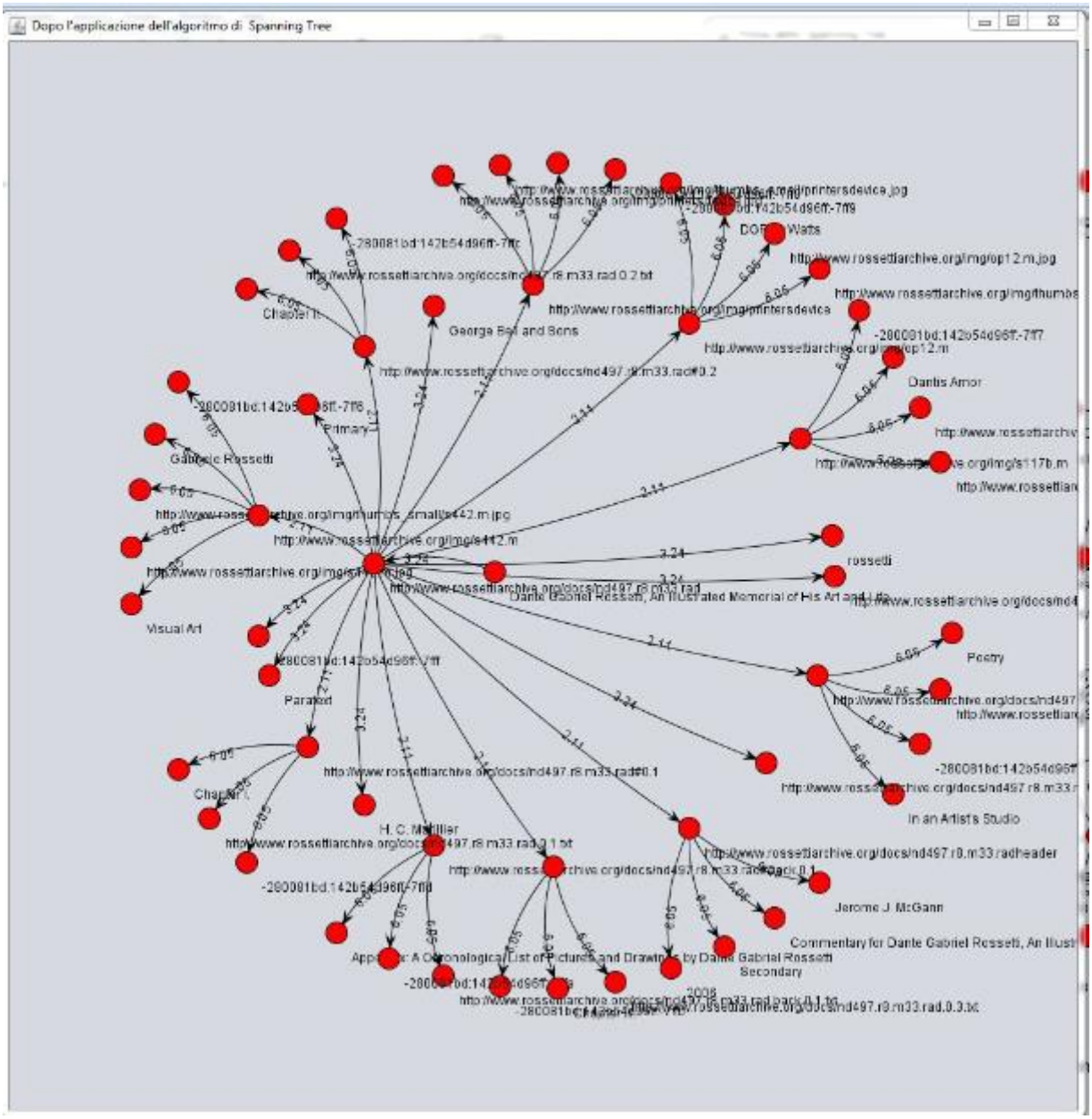

Figura 26 : Grafo ottenuto dopo l'applicazione dello Spanning Tree.

La figura precedente rappresenta l'albero ottenuto dopo l'applicazione dello spanning tree. Questo grafo contiene 57 nodi e 56 archi. L'algoritmo di Prim che è spiegato nella sessione "Teoria dei grafi" di questo documento è stato usato per ottenere il grafo finale.

Salvataggio del risultato dello spanning tree in un file XML

Dopo aver ottenuto lo spanning tree risultato, il problema che si poneva era di salvare il risultato in formato XML. Come visto all'inizio di questo capitolo, il formato XML permette è adatto per il salvataggio di strutture complesse (alberi, grafi…).

Per scrivere il documento RDF risultato, abbiamo quindi usato la seguente convenzione:

- Ogni tag(etichetta) XML può corrispondere ad un nodo o ad un arco.
- Ogni nodo ha un attributo "name"(il nome del nodo).
- Ogni arco ha due attributi "name" e "weight"(il nome ed il peso).
- Nella gerarchia di ogni nodo, possiamo aggiungere dei nodi e degli archi.
- Per fare il legame tra due nodi, si prende il nodo(tag node), l'arco che lo precede(tag edge) e lo leghiamo al padre(tag node). Quindi, tutti i tag edge hanno come padre un tag node che corrisponde all'origine dell'edge.

Una classe è stata realizzata per mettere in pratica la convenzione appena descritta.

Riportiamo di seguito il file XML generato dal nostro programma a partire dall'albero riportato in Figura 26.

```
<spanningTree>
 −<node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad#0.2">
    <edge name="http://purl.org/dc/elements/1.1/date" weight="6.059089762330736"/>
    <node name="-525c45af:1426665aa7a:-7fe8"> </node>
    <edge name="http://purl.org/dc/elements/1.1/title" weight="6.059089762330736"/>
    <node name="Chapter II."> </node>
    <edge name="http://www.collex.org/schema#text" weight="6.059089762330736"/>
    <node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad.0.2.txt"> </node>
    <edge name="http://purl.org/dc/terms/hasPart" weight="1.9770514548199751"/>
  −<node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad">
      <edge name="http://purl.org/dc/terms/isPartOf" weight="1.9770514548199751"/>
    −<node name="">
        <edge name="http://purl.org/dc/elements/1.1/date" weight="6.059089762330736"/>
        <node name="-525c45af:1426665aa7a:-7fe7"> </node>
        <edge name="http://www.collex.org/schema#text" weight="6.059089762330736"/>
        <node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad.0.3.txt"> </node>
        <edge name="http://purl.org/dc/elements/1.1/title" weight="6.059089762330736"/>
        <node name="Chapter III."> </node>
      </node>
      <edge name="http://purl.org/dc/terms/hasPart" weight="1.9770514548199751"/>
    −<node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad#0.1">
        <edge name="http://www.collex.org/schema#text" weight="6.059089762330736"/>
        <node name="http://www.rossettiarchive.org/docs/nd497.r8.m33.rad.0.1.txt"> </node>
        <edge name="http://purl.org/dc/elements/1.1/date" weight="6.059089762330736"/>
        <node name="-525c45af:1426665aa7a:-7fe9"> </node>
        <edge name="http://purl.org/dc/elements/1.1/title" weight="6.059089762330736"/>
```

Figura 27 : Esempio di output XML

Il file XML riportato in Figura 27 è la rappresentazione dell'albero di supporto in formato XML ottenuta applicando la convenzione descritta precedentemente.

Descrizione dell'applicazione

L'applicazione che è stata realizzata durante questo lavoro è un'applicazione molto semplice da utilizzare. Permette in pochi click di effettuare tutte le operazione su descritte; caricare un file RDF, scegliere il filtro da applicare, generare il grafo su cui è stato applicato l'algoritmo di Spannig Tree e di poter esportare il grafo in formato XML.

La figura seguente rappresenta la pagina principale dell'applicazione.

Figura 28 : Pagina principale dell'applicazione.

Come si vede dalla pagina figura precedente, dalla pagina principale, si può scegliere il file di cui si vuole estrarre l'albero (file RDF). Si potrebbe anche decidere se usare il dizionario RDF come filtro, o scegliere un altro file come filtro. Il bottone "Parsing" presente sull'interfaccia grafica permette di generare il grafo corrispondente al documento RDF, ed anche di visualizzare l'albero ottenuto dopo l'applicazione dello spanning tree sul grafo.

Per quanto riguarda il salvataggio, dopo aver fatto il parsing ed applicato lo spanning tree, possiamo salvare il file XML che descrive l'albero. La figura seguente rappresenta la pagina di salvataggio.

Figura 29 : Pagina di salvataggio dei risultati in formato XML.

Dalla figura precedente, si vede che si può esportare il risultato in formato XML tramite l'interfaccia grafica realizzata.

Conclusione

In questo lavoro, abbiamo presentato l'importanza del formato di dati RDF per la gestione di risorse appartenenti a diverse entità.

Prima di tutto, abbiamo presentato il contesto del nostro lavoro. In una seconda fase, abbiamo studiato i legami che esistono tra i dati RDF e le basi di dati relazionali; questo ci ha permesso di esplorare i lavori che sono stati fatti fino ad oggi sull'argomento.

Successivamente, ci siamo interessati ai vari algoritmi che si possono applicare sui grafi per ricavare il percorso ottimale di navigazione tra le differenti risorse contenute in un documento RDF.

Questo lavoro ci ha permesso di capire le numerose sfide che esistono oggi nella gestione di dati RDF, e soprattutto la necessità di mettere insieme le nostre conoscenze in vari ambiti (page ranking, sviluppo in Java, teoria dei grafi) per risolvere dei problemi informatici.

Bibliografia

1. **Damiani, Ernesto.** Protocollo di Spanning Tree.

2. **Byrne, Kate.** *Relational Database to RDF Translation in the Cultural Heritage Domain.*

3. **Chittayasothorn, Wajee Teswanich and Suphamit.** *A Transformation from RDF Documents and Schemas.*

4. **InfoLab, Stanford.** The PageRank Citation Ranking: Bringing Order to the Web.

5. **Pasquinelli, Matteo.** *L'algoritmo PageRank di Google: diagramma del capitalismo cognitivo e rentier dell'intelletto comune.*

6. **Gagnon, Michel.** *Brève introduction à RDF.*

7. **Manchester, The University of.** *An Introduction to RDF(S) and a Quick Tour of OWL.*

8. **STEFAN DECKER, PRASENJIT MITRA, AND SERGEY MELNIK.** *Framework for the Semantic Web:An RDF Tutorial.* Dicember 2000.

9. Apache Jena. *http://jena.apache.org/.* [Online]

10. Java Universal Network/Graph Framework. *http://jung.sourceforge.net.* [Online]

11. JAMA : A Java Matrix Package. *http://math.nist.gov/javanumerics/jama/.* [Online]

12. **Wikipedia.** http://wiki.collex.org/index.php/RDF_samples. [Online]