UNIVERSITA' DEGLI STUDI
DI MODENA E REGGIO EMILIA

Facolta' di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea Triennale

Tecniche per l'Interrogazione in Linguaggio Naturale di Dati Modellati a Grafo

Relatore:
Prof. Riccardo Martoglia

Candidato:
Alice Messori

Anno Accademico 2009/2010

Sommario

Introduzione

Osservando le tecnologie attuali si nota immediatamente come i modelli di dati basati su strutture a grafo si stiano diffondendo sempre di piu' in numerose aree d'applicazione legate alle basi di dati, basti pensare ai database biologici e chimici o ai dati presenti sul Web. Tali strutture risultano essere particolarmente avvezze alla manipolazione di grandi quantita' di dati in quanto permettono di schematizzare un'ampia varieta' di situazioni e di processi, rendendone piu' semplice l'analisi qualitativa e permettendo un elevato grado di espressivita'.

In contesti come questi, dove si gestiscono elevate quantita' di dati eterogenei, e' quasi impensabile avere una completa conoscenza del vocabolario utilizzato e delle modalita' di organizzazione dei dati stessi. E' quindi necessario sfruttare e sviluppare meccanismi che possano permettere un'interrogazione flessibile delle collezioni di dati.

Un altro tassello importante che ha portato allo sviluppo di questo elaborato e' costituito dal processo che interessa il mondo dell'informatica ed il rapporto con le tecnologie nel periodo attuale; le applicazioni software e l'importanza che la tecnologia riveste nella nostra vita quotidiana stanno portando i tipi piu' disparati di utenti e persone a fare uso di tecnologie software di ogni tipo.

In questi contesti risulta evidente come le tecnologie debbano sempre piu' avvicinarsi alle esigenze degli utenti, indipendentemente dal loro grado di formazione scientifica-tecnica specifica, ed indipendentemente dall'applicazione che stanno utilizzando.

Una delle sfide piu' ambiziose ed importanti della ricerca in questo ambito e' lo sviluppo di meccanismi di interrogazione flessibile dei dati che consentano agli utenti di esprimere facilmente le loro richieste e recuperare agevolmente i dati utili.

GeX (*Graph EXplorer*) e' un software che permette l'interrogazione approssimata di dati modellati a grafo, ovvero fa si che gli utenti possano interrogare un determinato dataset senza necessariamente conoscere il vocabolario dei dati o come essi siano organizzati.

Nonostante questo pero' persiste il problema della costruzione dell'interrogazione: gli utenti sarebbero infatti tenuti, per interrogare efficientemente le collezioni di dati, a conoscere le modalita' e i linguaggi specifici attraverso cui costruire interrogazioni anche se queste sono flessibili e permettono loro di astrarre dai dettagli della struttura dati.

L'obiettivo ambizioso che ci si pone e' quindi quello di mettere l'utente nella condizione di interrogare le collezioni di dati tramite il linguaggio naturale, la lingua che si parla ogni giorno.

Sfruttando Boxer, un software che esegue l'analisi grammaticale e morfologica di frasi in linguaggio naturale per restituirne poi una rappresentazione formale basata su grafi, e la flessibilita' messa a disposizione da GeX ci proponiamo di rintracciare dei meccanismi univoci e ben definiti che ci permettano di trasformare interrogazioni in linguaggio naturale in interrogazioni utilizzabili da un software.

Risulta evidente che uno degli ostacoli principali e' rappresentato dalla grande varieta' di termini e strutture linguistiche che si trovano nel linguaggio naturale e che gli utenti possono utilizzare, ad esempio l'abitudine a sottintendere termini, usare sinonimi o strutture linguistiche particolari. Ci proponiamo quindi di rintracciare dei meccanismi utilizzabili in modo generale ed indipendente dal linguaggio naturale.

In questo elaborato sono presenti quattro capitoli.

Il primo capitolo contiene un'analisi degli standard XML ed RDF su cui si basano le entita' fondamentali della tesi, rispettivamente le interrogazioni flessibili che GeX esegue e le collezioni di dati.

Il secondo capitolo contiene un'attenta analisi e presentazione delle strutture dati modellate a grafo e delle tecniche per l'interrogazione approssimata di grafi implementate grazie al software GeX e al parser Boxer.

Il terzo capitolo, cuore della tesi, presenta il lavoro svolto nella ricerca delle strutture che permettono di trasformare i grafi linguistici generati da Boxer in grafi flessibili utilizzabili da GeX per l'interrogazione di dati.

L'ultimo capitolo contiene le prove sperimentali effettuate su due differenti collezioni di dati con l'obiettivo di verificare la correttezza dei pattern rintracciati, ma anche di capire quali sono i limiti e le potenzialita' dell'interrogazione flessibile di dati in linguaggio naturale. Inoltre verra' presentata un'analisi dei risultati raccolti nel corso della sperimentazione.

Parte I

Introduzione allo studio

Capitolo 1

Standard

All'interno di questo capitolo verranno presentati gli standard che sono stati alla base dello studio del problema affrontato dalla tesi. Come primo standard viene presentato XML (eXtendible Markup Language) utile per la definizione di nuovi linguaggi di markup; successivamente verranno esposte le caratteristiche del modello RDF (Resource Description Framework) usato per rappresentare i metadati connessi ad una risorsa Web tramite l'utilizzo di un linguaggio che descriva la semantica della risorsa stessa.

1.1. XML (eXtensible Markup Language)

La sigla XML (*eXtensible Markup Language*) identifica un metalinguaggio estendibile di markup sviluppato da W3C [1] (*World Wide Web Consortium*) e deriva, al pari di HTML, da SGML (*Standard Generalized Markup Language*), primo vero linguaggio di markup risalente agli anni '70.

SGML fu utilizzato da molte imprese importanti, quali IBM e il Dipartimento della Difesa; data la sua elevata complessita', a partire dalla

fine degli anni ottanta, W3C decise di sviluppare un linguaggio di markup piu' semplice e personalizzabile: nacque cosi' XML.

Come dice il nome stesso, XML e' un linguaggio di markup il cui compito e' quello di definire altri linguaggi; fondamentale e' la caratteristica di estendibilita': infatti XML, a differenza di HTML, non basa la sua grammatica su un set fissato di tag per la descrizione e la formattazione di pagine web e ipertesti, ma e' un metalinguaggio senza tag predefiniti, che permette di specificare tag personali per creare metalinguaggi per la descrizione di documenti strutturati.

Passiamo ora a descrivere nel dettaglio la grammatica e la struttura dei documenti XML.

I documenti di tipo XML si presentano come file di solo testo, il che li rende completamente indipendenti dalla piattaforma.

La base della grammatica XML, al pari dell'HTML, si fonda sul concetto di *tag*, marcatori il cui compito e' quello di dare una semantica al testo. I tag iniziano con un carattere "<" (parentesi angolata aperta), finiscono con ">" (parentesi angolata chiusa) e contengono una stringa di caratteri (esempi di tag HTML sono <HEAD>, <BODY>, </B> ecc).

Nell'XML invece i nomi dei tag sono assolutamente personalizzabili e ogni elemento e' composto da un tag di apertura, il rispettivo tag di chiusura ed eventualmente un contenuto tra i due tag:

<nome>Eco</nome>

E' anche possibile definire elementi vuoti utilizzando tag vuoti:

<nome/>

L'XML, a differenza dell'HTML, e' molto rigido riguardo alla sintassi dei suoi documenti: i tag non possono iniziare con numeri o caratteri speciali e non possono contenere spazi; i nomi dei tag sono case-sensitive; i tag devono essere bilanciati, cioe' ogni tag aperto deve sempre essere chiuso, e

deve essere rispettato l'ordine di annidamento, cioe' l'ultimo tag aperto dev'essere il primo ad essere chiuso.

Rispettando queste regole si arriva alla stesura di un documento XML *well-formed* (ben formato) dove sono rintracciabili le seguenti caratteristiche:

- un *prologo*, prima istruzione che appare scritta nel documento XML; nel nostro caso: *<?xml version="1.0" encoding=" UTF-8"?>* dove nella prima parte e' specificata la versione XML che si utilizza, nella seconda i caratteri utilizzati.
- un unico *elemento radice* (ovvero il nodo principale, *root element*) che contiene tutti gli altri nodi del documento;
- i *tag* del documento sono tutti bilanciati e ben annidati.

Facciamo un esempio.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Documento XMl contenente informazioni su animali domestici -->
<animale>
        <gatto>
            <nome>Eco</nome>
            <eta>6 mesi</eta>
            <razza>non pervenuta</razza>
            <manto>
                <colore_primario>marrone</colore_primario>
                <colore_secondario>nero</colore_secondario>
                    <disegno>tigrato</disegno>
            </manto>
        <gatto>
</animale>
```

Come si puo' vedere questa e' un brano di documento XML ben formato: tutti i tag aperti sono chiusi rispettando l'annidamento, inoltre il formato dei tag e' corretto.

Ecco un esempio di documento non ben formato:

```
<?xml version="" encoding="UTF-8"?>
<!-- manca la versione nell'intestazione -->
<animale>
        <gatto>
            <nome>Eco</NOME>
            <eta>6 mesi
            <razza>non pervenuta</razza></eta>
            <manto>
                <colore_primario>marrone</colore_primario>
                <colore_secondario>nero</colore_secondario>
                    <disegno>tigrato</disegno>
            </manto>
        <gatto>
</animale>
```

Ciascun elemento puo' avere inoltre, all'interno del tag di apertura, un numero arbitrario di coppie *attributo = valore* che permettono di classificare ulteriormente l'informazione contenuta nel tag:

<peso unita_misura = "kg">3</peso>

Un documento XML e' un albero ordinato in cui i nodi interni rappresentano gli elementi stessi, i tag, mentre i nodi foglia sono i dati veri e propri.

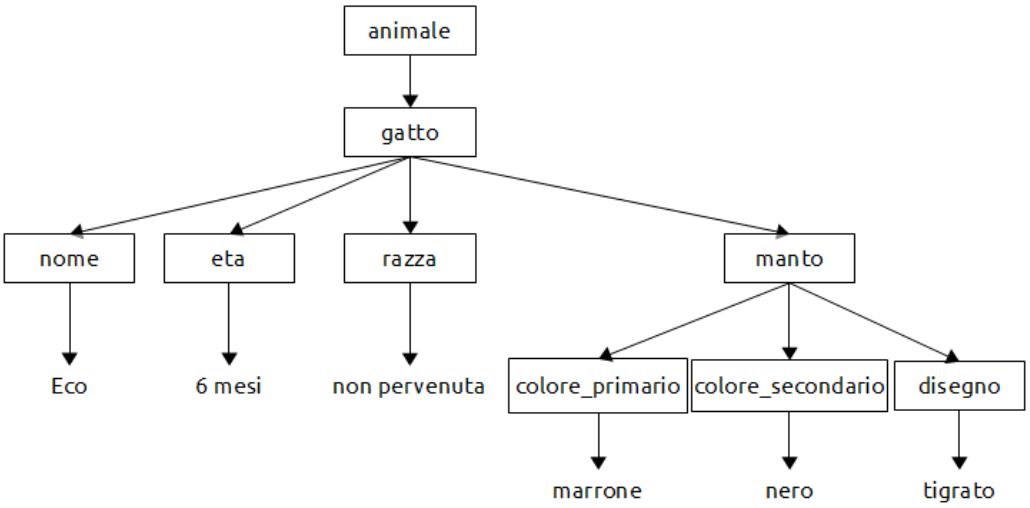


Figura 1: rappresentazione a grafo di un file XML

E' importante sottolineare che XML non fa elaborazioni ma sono i parser che si occupano di rendere disponibili i dati per le applicazioni che li richiedono; infatti questi strumenti vanno a recuperare e verificare i file XML ed il loro contenuto e lo passano alle applicazioni. Maggiori dettagli nel paragrafo successivo.

Utilizzi.

Il linguaggio XML e' fondamentale nell'ambito di questa tesi in quanto le query sottoposte a GeX sono scritte secondo la sua sintassi.

1.2. RDF (Resource Description Framework)

Il modello RDF (*Resource Description Framework* = "Struttura per la descrizione di risorse") e' sviluppato da W3C [2] (*World Wide Web Consortium*) per descrivere i metadati (le informazioni) connessi ad una risorsa Web, tramite l'utilizzo si un linguaggio che descriva la semantica della risorsa stessa.

RDF e' un modello completamente astratto e puo' essere rappresentato in modi e forme diverse (ad esempio tramite XML, OWL ecc.); inoltre questo strumento e' indipendente dal dominio applicativo che ne fa uso: il suo scopo e' quello di essere in grado di descrivere le informazioni relative ad un qualsiasi domino.

Riassumendo, RDF e' uno strumento utile nelle situazioni in cui le informazioni connesse alle risorse devono essere elaborate da applicazioni, piuttosto che visualizzate da degli utenti, e fornisce un quadro comune per esprimere tali metadati in modo che sia possibile il loro scambio tra le applicazioni senza perdita di significato.

Il modello RDF si basa su due componenti: *RDF Model and Syntax* e *RDF Schema*.

La prima componente riguarda il modello dei dati (ovvero la struttura del modello RDF) tramite cui descrivere le risorse, e definisce la sintassi di questo modello (ad esempio XML, portando cosi' alla dicitura "RDF/XML"); *RDF Schema* invece descrive la sintassi per definire gli schemi e i vocaboli dei metadati.

Passiamo ora a descrivere l'RDF Data Model. Esso si basa su tre unita' atomiche fondamentali: *risorse, proprieta'* e *valori*.

- *Risorsa:* il modello RDF definisce ogni oggetto o cosa come *risorsa*. Tali risorse sono rappresentabili tramite un URI (*Universal Resource Identifier*, acronimo piu' generico rispetto a "URL") una stringa che funge da identificatore univoco. Queste risorse possono essere elementi del web (immagini, indirizzi Web, file,

servizi ecc.) o elementi fisici che non appartengono direttamente al Web ma che sono rintracciabili tramite esso (es: una persona fisica, un oggetto messo in vendita tramite siti di e-commerce).

- *Proprieta':* le *proprieta'* specificano le caratteristiche delle risorse. Esse possono essere proprieta', attributi o relazioni utilizzate per descrivere una risorsa; sono definite anch'esse da URI e legano le risorse a dei valori. Il significato e le caratteristiche di questo componente sono definite nell'*RDF Schema*.
- *Valore:* un *valore* e' un tipo di dato primitivo (ad esempio una stringa contenente un URI, un valore numerico ecc.)

L'unita' di base per la rappresentazione di un'informazione RDF e' lo *statement*: si tratta di una frase, o asserzione, caratterizzata da una struttura fissa i cui elementi di base sono un *soggetto* (una risorsa), un *predicato* (una proprieta') e un *oggetto* (una risorsa o un letterale). Questi elementi vanno a comporre una *tripla* nella forma *(soggetto, predicato, oggetto)* che graficamente puo' essere rappresentata tramite un grafo orientato in cui i nodi rappresentano le risorse (ovvero i soggetti) o i valori primitivi (l'oggetto) mentre gli archi sono etichettati con le proprieta' (i predicati):

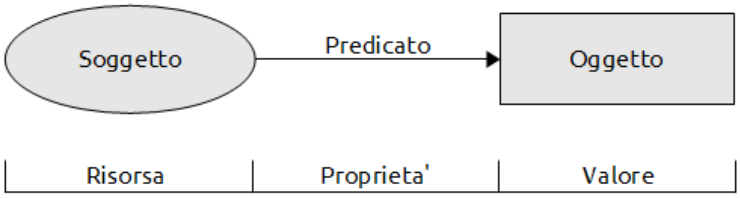


Figura 2: rappresentazione grafica delle triple RFD

Da un punto di vista grafico solitamente gli ellissi contengono le risorse, mentre i rettangoli i valori di tali risorse in base alla proprieta' specificata sull'arco. Il nodo di partenza e' il soggetto della tripla RDF.

Esempio di statement RDF

Prendiamo in considerazione una semplice frase in linguaggio naturale:

Il sito www.esempio.com ha come creatrice Arianna Bianca.

Per quanto detto fino ad ora possiamo ricercare i tre componenti fondamentali di una tripla RDF:

www.esempio.com	**ha_creatore**	**Arianna Bianca**
Soggetto (= risorsa)	Predicato (= proprieta')	Oggetto (= valore)

Leggendola in altri termini: la proprieta' "ha_creatore" della risorsa "www.esempio.com" ha valore "Arianna Bianca".

Graficamente:

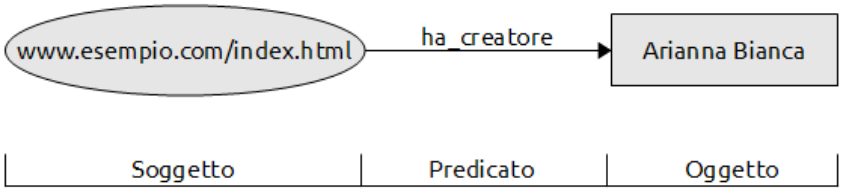


Figura 3: rappresentazione grafica della frase d'esempio.

RDF permette di espandere a piacimento i grafici in quanto i valori delle entita' oggetto possono essere considerati risorse, ovvero soggetti, di nuove triple, permettendo cosi' di rappresentare situazioni anche molto complesse ed articolate.

Possiamo quindi espandere il nostro esempio come segue:

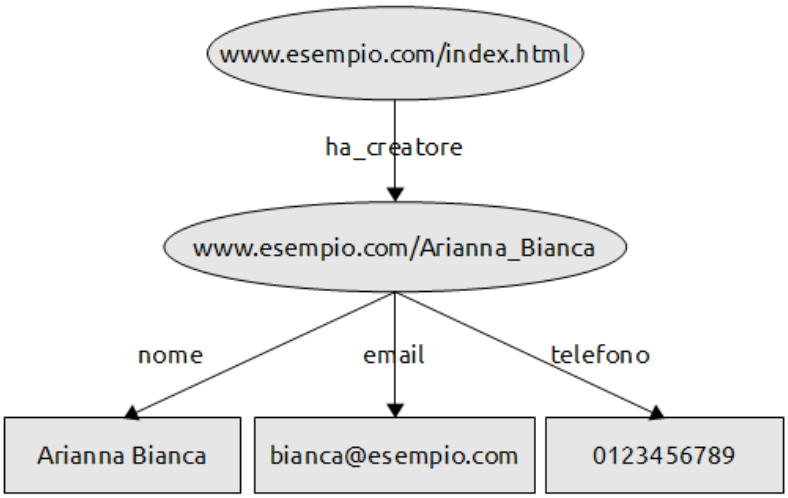


Figura 4: estensione del grafo in Figura 3

Come precedentemente accennato RDF e' un modello astratto e per esprimere al meglio gli statement e' previsto l'uso di una sintassi basata su XML; la Figura 3 puo' quindi essere tradotta come segue:

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:a="http://esempio.com/schema_autore">
        <rdf:Description about="http://esempio.com/index.html">
                <a:ha_creatore>
                        Arianna Bianca
                </a:ha_creatore>
        </rdf:Description>
</rdf:RDF>
```

L'elemento *<rdf:RDF>* racchiude la definizione dello statement RDF ed al suo interno troviamo la definizione di due Namespace: il primo e' relativo al Namespace RDF, mentre il secondo contiene l'URI che identifica lo schema RDF utilizzato per descrivere la semantica e le convenzioni che regolano l'utilizzo delle proprieta' presenti nello statement.

La descrizione del metadato e' contenuta all'interno dell'elemento *<rdf:Description>* ed il suo attributo *about* identifica la risorsa alla quale si riferisce il metadato stesso.

La proprieta' dello statement e' descritta utilizzando il tag *<a:ha_creatore>*, secondo le regole che sono espresse nel relativo schema RDF.

Riportiamo ora il codice RDF/XML corrispondente al grafo in Figura 4:

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:a="http://esempio.com/schema_autore">
        <rdf:Description about="http://esempio.com/index.html">
                <a:ha_creatore
        rdf:resource="http://esempio.com/Arianna_Bianca">
        </rdf:Description>

        <rdf:Description about="http://esempio.com/Arianna_Bianca">
                <a:nome>Arianna Bianca</a:nome>
                <a:email>bianca@esempio.com</a:email>
                <a:telefono>0123456789</a:telefono>
        </rdf:Description>
</rdf:RDF>
```

Come si vede, in questo secondo esempio vengono definiti due risorse (identificate entrambe dal rispettivo *<rdf:Description>*) che sono messe in relazione tramite l'attributo *rdf:resource* presente in *<a:ha_creatore>*; ne deriva che la descrizione della seconda risorsa (quella relativa ai dati dell'autore del sito) viene assegnata come valore della proprieta' *ha_creatore* della prima risorsa.

Utilizzi:

Per quanto riguarda l'ambito di questa tesi il linguaggio RDF/XML e' utilizzato nei file contenenti i dati su cui eseguire le interrogazioni, ovvero i file che rappresentano l'insieme di dati del nostro dominio, file che viene indicizzato nella fase iniziale da GeX e caricato all'interno del database.

Capitolo 2

Tecniche per l'interrogazione approssimata di grafi

In questo capitolo verranno presentate le tecniche per l'interrogazione approssimata di grafi e le strutture dati utilizzate per sviluppare la tesi. Gli obiettivi che si perseguono sono quello di mostrare le caratteristiche e le potenzialita' dello schema dei dati e successivamente descrivere le funzionalita' dei due software utilizzati: GeX e Boxer.

2.1. Dati modellati a grafo

Al giorno d'oggi i modelli di dati basati su strutture a grafo sono in forte espansione in diverse aree applicative, basti pensare all'enorme quantita' di dati presenti sul Web o ai database biologici e chimici. In questi ambiti e' quindi di largo uso il modello RDF di rappresentazione di metadati, nato proprio per modellare facilmente grandi insiemi di dati all'interno di un contesto in cui i dati vengono scambiati e manipolati da diverse applicazioni, ma senza che questo ne comporti perdita di significato.

Per cominciare chiariamo il concetto di *grafo*. Con questo termine si indicano insiemi di elementi discreti in cui coppie di oggetti sono connesse tramite link [3].

In termini piu' informali per grafo si intende una struttura costituita da:

- oggetti semplici chiamati *vertici* o *nodi*
- collegamenti tra i vertici, detti *link* o *archi*.

A seconda dei tipi di collegamenti che si instaurano tra i nodi di un grafo possiamo avere grafi diretti (o orientati), in cui e' specificato il verso del collegamento, o indiretti (non orientati), senza nessuna specifica.

Generalmente un grafo viene rappresentato come un insieme di cerchi, contenenti il nome o identificativo del nodo, e segmenti o curve che collegano tali cerchi.

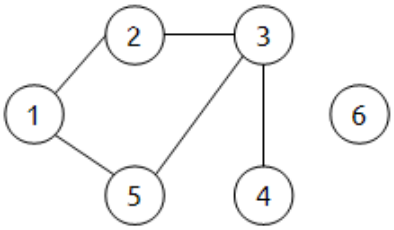


Figura 5: grafo non orientato con 6 nodi e 5 archi

Questo tipo di struttura dati permette di schematizzare una grande varieta' di situazioni e di processi, rendendone piu' semplice l'analisi qualitativa.

In ambito matematico il loro studio, la Teoria dei Grafi [link], costituisce un'importante parte della combinatoria; i grafi inoltre sono utilizzati in aree come topologia, teoria degli automi, funzioni speciali,

geometria dei poliedri; queste strutture si incontrano anche in vari ambiti dell'informatica, ad esempio per schematizzare programmi, circuiti, reti di computer, mappe di siti. Essi inoltre sono alla base di modelli di sistemi e processi studiati nell'ingegneria, nella chimica, nella biologia molecolare, nella ricerca operativa, nella organizzazione aziendale, nella geografia (sistemi fluviali, reti stradali, trasporti), nella linguistica strutturale, nella storia (alberi genealogici, filologia dei testi).

La modellazione di insiemi di dati tramite la tecnica dei grafi risulta quindi molto conveniente in contesti in cui le connessioni tra gli elementi del dataset rivestono un ruolo dominante. Ne deriva che questo modello si e' diffuso anche all'interno del contesto delle basi di dati dove i *database a grafo*, che sfruttano nodi e archi per rappresentare e archiviare le informazioni, affiancano il classico modello relazionale, che si avvale di tabelle.

Infatti i database a grafo risultano essere piu' veloci degli schemi relazionali nell'associazione di set di dati, e mappano piu' direttamente le strutture di applicazioni orientate agli oggetti; scalano piu' facilmente a grandi quantita' di dati e non richiedono le tipiche e costose operazioni di unione del modello relazionale; inoltre dipendono meno da un rigido schema ER (*Entity-Relation*) e risultano essere piu' adeguati per gestire dati mutevoli con schemi evolutivi. Al contrario, i database relazionali sono tipicamente piu' veloci nell'eseguire le stesse operazioni su un grande numero di dati.

I set di dati utilizzati durante lo sviluppo della tesi si collocano proprio all'interno di questo contesto; la prima raccolta di dati descrive le caratteristiche geofisiche degli Stati Uniti d'America ed e' modellato secondo la struttura a grafo del modello RDF ed e' stata utilizzata durante tutto lo sviluppo del lavoro.

Di seguito viene riportato lo schema dei dati:

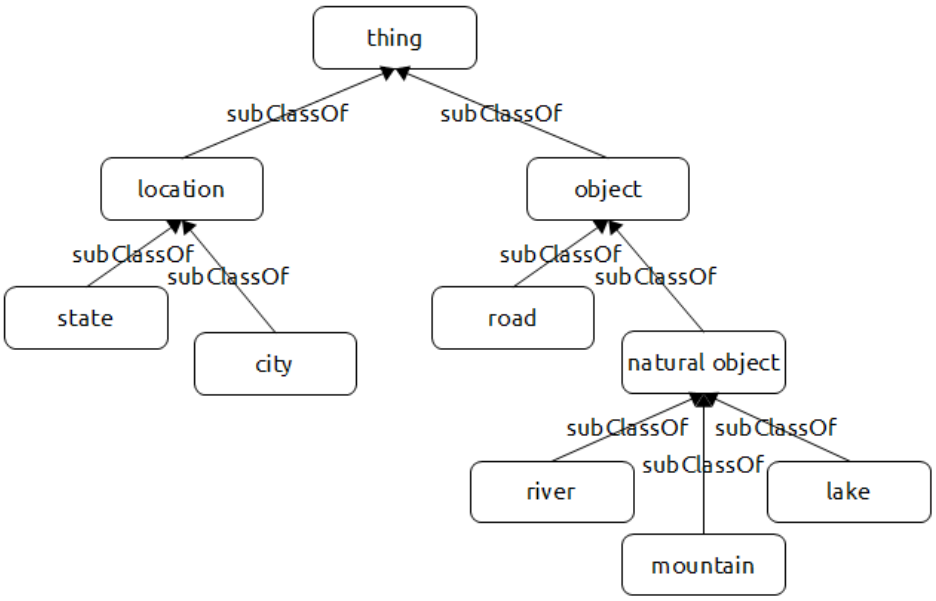


Figura 6: schema dei dati di geobase

Ogni classe e' ulteriormente specificata e connessa ad una serie di proprieta'; per chiarezza riportiamo solo le proprieta' legate alla classe "state":

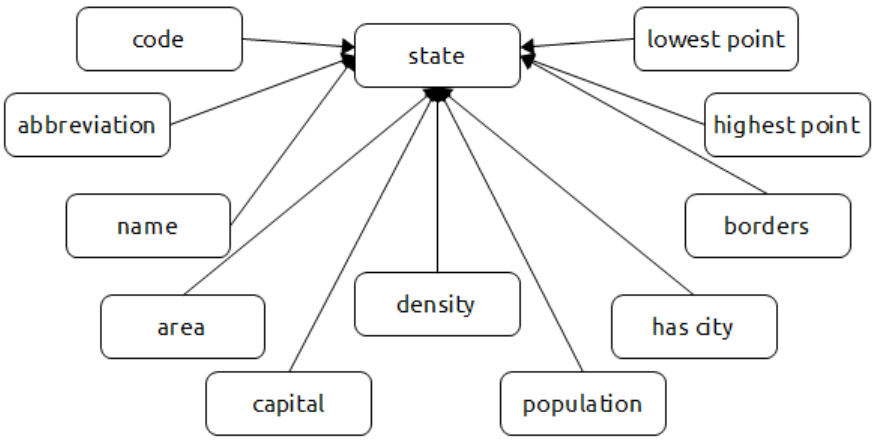


Figura 7: piccola porzione del proprieta' di una classe

Come indicato al capitolo 1, paragrafo 1.2 relativa allo standard RDF, le proprieta' hanno il compito di connettere gli oggetti (per esempio uno stato) a specifici valori (ad esempio la sua capitale). Ecco quindi un esempio

concreto di una piccola parte del livello delle istanze del nostro archivio geobase:

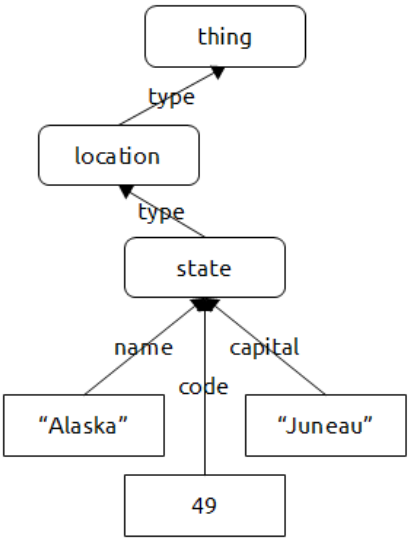


Figura 8: piccolo dettaglio del livello delle istanze

La seconda collezione di dati e' stata utilizzata solo nell'ambito dell'ultima parte della tesi, ovvero nel momento in cui si e' voluto testare ulteriormente il lavoro svolto; si tratta di una collezione di bibliografica di tipo DBLP. Data la grande dimensione dell'intera bibliografia se ne e' utilizzata una parte ristretta contenente i dati relativi allo schema sotto proposto:

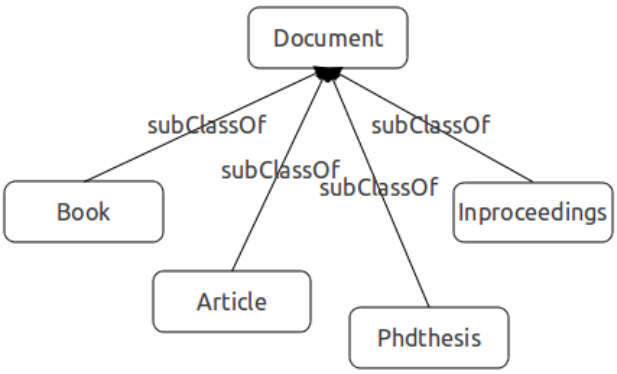


Figura 9: schema dei dati della collezione DBLP

Come nel caso di geobase ogni classe e' connessa ad un gran numero di proprieta', di seguito quelle della classe Book.

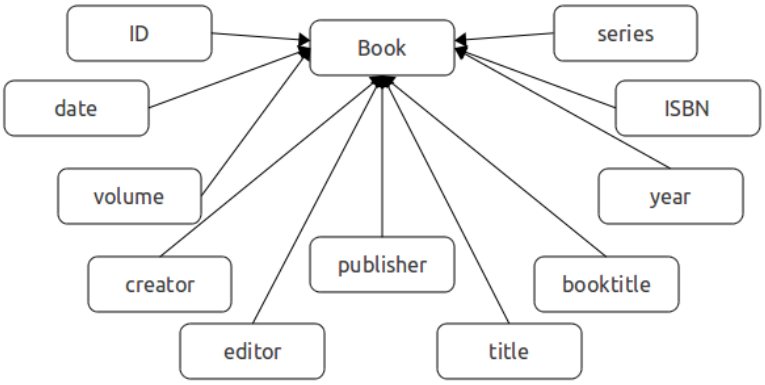


Figura 10: dettaglio delle proprieta' della classe Book

Infine un piccolo esempio del livello delle istanze:

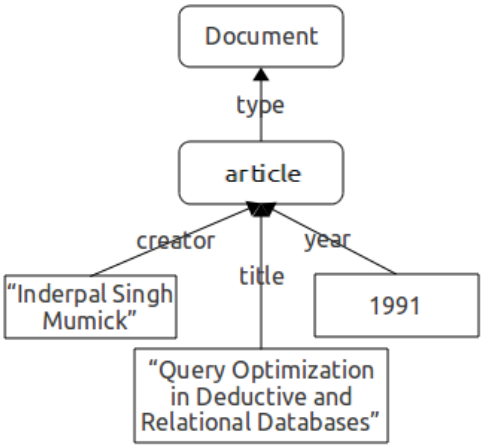


Figura 11: dettaglio del livello delle istanze di DBLP

Dall'osservazione degli schemi presentati risulta evidente che i file RDF completi sono molto ampi e complessi dato il gran numero di dati e connessioni che si vengono a creare tra essi.

L'operazione fondamentale che si desidera fare su un insieme di dati di questo tipo e' l'interrogazione. L'obiettivo dell'interrogazione e' quello di estrarre dal dataset i dati e le informazioni di interesse che rispettano i vincoli imposti dalla query. Solitamente il linguaggio con cui interroghiamo un dataset dipende fortemente dal tipo di modello utilizzato per la sua organizzazione; ad esempio se organizziamo i dati tramite un modello relazionale saremo portati a costruire le interrogazioni con un linguaggio come SQL; dall'altra parte, organizzando i dati secondo un modello a grafo, il linguaggio della query potrebbe essere SPARQL, linguaggio di l'interrogazione per dataset di tipo RDF.

Le query costruite secondo questi linguaggi hanno una struttura rigida e precisa e risultano ben formate e non ambigue; di seguito due esempi di query in linguaggio SQL legati all'archivio geobase; ipotizziamo che i dati siano suddivisi in tabelle a seconda della classe d'appartenenza (city, road, state, river etc.).

"Quali sono le citta' del South Dakota?"

```
SELECT city.*
        FROM city
        WHERE city.inState == "South Dakota"
ORDER BY city.name;
```

La risposta a questa query sara' l'elenco delle citta' che si trovano nello stato del South Dakota, con le rispettive caratteristiche, in ordine alfabetico per nome.

"Le caratteristiche degli stati attraverso cui passa una strada con numero 79"

```
SELECT  state.*
        FROM road, state
        WHERE road.passesThrough == state.name
                AND road.number == 79
ORDER BY state.name;
```

In questo caso abbiamo un'interazione tra due tabelle, unite tramite una join espressa dalla condizione "*road.passesThrough == state.name*".

Da questi esempi risulta chiaro che interrogare in modo piu' approfondito e complesso questo tipo di schema di dati tramite un linguaggio "classico" come SQL o SPARQL risulta molto complesso: cio' e' dovuto sia alla complessita' nella costruzione della query, sia alle conoscenze che l'utente puo' avere dello schema dei dati e del suo vocabolario.

In seguito a queste considerazioni si e' fatta strada l'idea che l'implementazione di un meccanismo di interrogazione flessibile basata su grafi sia la soluzione da perseguire maggiormente.

2.2. Interrogazione approssimata di grafi - GeX

GeX (*Graph EXplorer*) e' un software che permette l'interrogazione approssimata di dati modellati a grafo, ovvero fa si che gli utenti possano interrogare un determinato dataset senza necessariamente conoscere il vocabolario dei dati o come essi siano organizzati [4, 5].

Il software e' stato sviluppato dall'isgroup dell'Universita' di Modena e Reggio Emilia in seguito ad alcune osservazioni sulla realta' attuale del mondo delle applicazioni: in diverse aree d'interesse i dati sono caratterizzati da informazioni semantiche modellate attraverso strutture basate su grafi. Questo tipo di modellazione e' molto diffuso in quanto permette un elevato grado di espressivita', ed e' anche utilizzato nel campo del Semantic Web dove XML e RDF si basano proprio sulla rappresentazione dei dati a grafo.

L'ampia eterogeneita' e complessita' dei dati e dei domini a cui essi fanno riferimento non facilitano l'estrazione di informazioni e rendono impraticabili la loro interrogazione attraverso classiche query che utilizzano linguaggi come SQL o SPARQL. E' infatti quasi impensabile che un utente (intendiamo un utente generico) abbia una conoscenza completa dello schema secondo cui i dati sono organizzati, del vocabolario usato, delle modalita' attraverso cui costruire query che soddisfino le loro richieste e via dicendo. In questo contesto la conclusione naturale a cui si arriva e' che, non potendo l'utente avere completa conoscenza dei dati e delle loro caratteristiche, e' necessario fare il passo opposto, ovvero permettere l'interrogazione flessibile dei dati tramite query flessibili, in modo da facilitare gli utenti nell'espressione delle loro richieste, anche se si tratta di richieste dove mancano informazioni o in cui alcune informazioni sono vaghe e non precise. Naturalmente queste interrogazioni flessibili devono partire dal linguaggio naturale dell'utente.

Per raggiungere tale obiettivo il punto di partenza e' quello di utilizzare un modello di query basate su parole chiave, in modo da astrarre

completamente dalla struttura della query; l'applicazione solo di questo modello risulta pero' povera, soprattutto in contesti in cui si lavora non solo sulla sintassi dei termini, ma anche sulla semantica del contesto e delle relazioni.

L'obiettivo che si vuole raggiungere e' quello di mettere gli utenti in una condizione tale da essere in grado di includere vari livelli della struttura nelle loro query, cosicche' possano specificare meglio le loro esigenze in base alla conoscenza parziale che possono avere della struttura dei dati.

In seguito a queste considerazioni e' stato quindi sviluppato GeX, un software in grado di supportare l'interrogazione approssimata di query eseguite su dati modellati a grafo in un'ampia gamma di scenari applicativi diversi.

Le principali caratteristiche di questo software includono:

- la *generalita'*: GeX permette di interrogare diversi tipi di dati modellati a grafo che differiscono sia per la struttura che per il modello adottato per la loro rappresentazione (ad esempio XML, RDF ecc);
- fornisce un linguaggio *semanticamente ricco* per la definizione delle query, permettendo cosi' all'utente di specificare le proprie richieste vaghe, attraverso l'espressione di diversi tipi di informazioni evasive e mancanti, cosi' come di precisare le condizioni dei vincoli sui dati;
- supporta la corrispondenza approssimativa delle query solo nel caso in cui tali corrispondenze derivino da approssimazioni significative. Per fare questo GeX sfrutta la nozione di relazione *Semantic Relatedness (relazione semantica)* secondo cui le connessioni tra le coppie di nodi in un grafo sono connessioni significative dal punto di vista semantico;
- implementa specifiche strutture dati e indici che supportano efficientemente i meccanismi per la ricerca di risposte

significative alle query. Tali strutture e indici sono efficientemente sfruttati da un algoritmo top-k che restituisce le k risposte piu' significative per la mia query.

Vediamo ora come lavora GeX.

Il software e' composto da tre moduli fondamentali:

- *DataExtractor*: modulo che si occupa di estrarre i dati presenti nei file RDF e di inserirli all'interno di un apposito Database;
- *PathExtractor*: modulo dedicato alla creazione dei percorsi di tipo SR, quindi collega i nodi in base a un concetto di connessione semantica;
- *MixedQuery*: modulo dedicato alla lettura delle query in formato XML e alla loro esecuzione. Questo modulo restituisce anche diverse statistiche sui tempi con cui GeX ha elaborato le k risposte.

Per analizzare meglio il funzionamento del software consideriamo una piccola porzione del file RDF geobase usato per lo sviluppo della tesi; nel grafo non sono riportati gli attributi relativi ai nomi degli oggetti perche' appesantirebbero troppo la struttura.

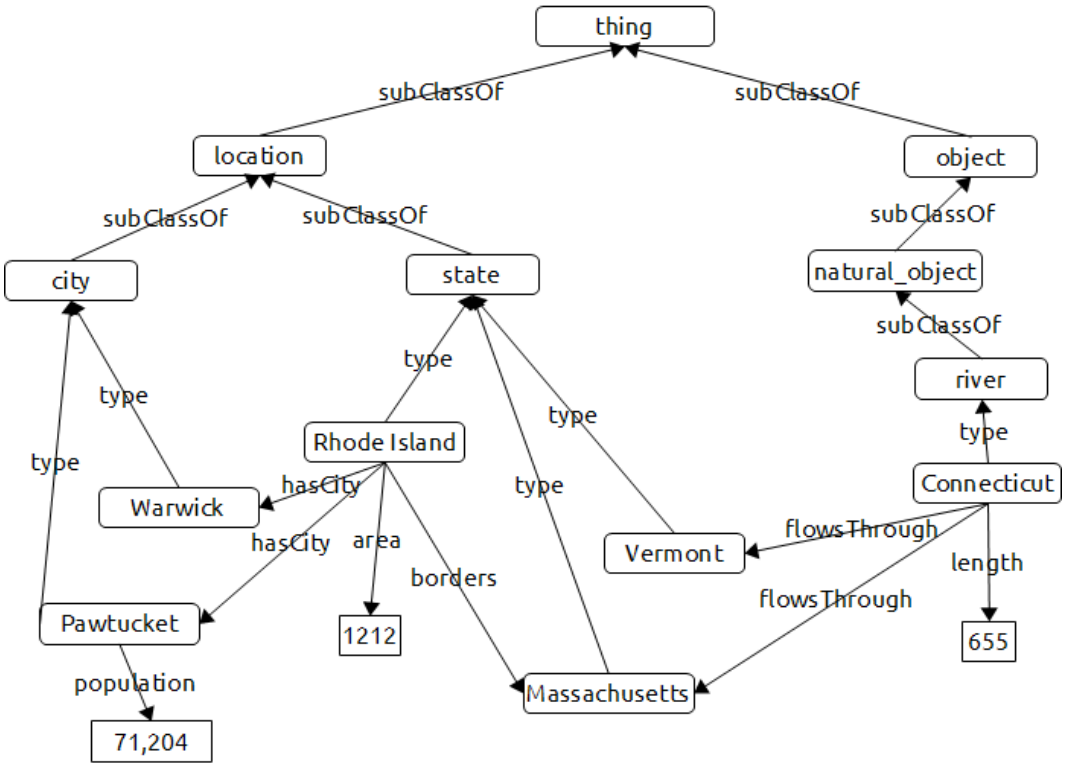


Figura 12: porzione del file geobase contenente schema e istanze del set di dati

Osservando questa piccola porzione di schema e istanze e per quanto detto al paragrafo precedente (2.1) risulta chiaro come le classiche interrogazioni in linguaggio SQL o SPARQL risultino inadeguate.

Riportiamo ora alcune interrogazioni tipo che un utente potrebbe voler eseguire osservando la figura:

Query 1: *"I luoghi che confinano con il Massachusetts e che hanno una citta' chiamata "Warwick"."*

Query 2: *"Gli stati che sono in relazione in qualche modo con un oggetto chiamato "Connecticut"."*

Query 3: *"Le informazioni relative a un luogo con un nome simile a "Pawtuc" e con una popolazione superiore ai 68.000 abitanti."*

Osservando le query risulta immediato come il linguaggio naturale e' si' comodo per l'utente, ma e' altrettanto complesso per la definizione di interrogazioni automatiche gestibili da un compilatore.

Facendo riferimento alle figura 9 estraiamo dallo schema RDF di geobase le porzioni di grafo che corrispondono alle nostre interrogazioni; per una migliore lettura abbiamo evidenziato le istanze che rappresentano i risultati delle interrogazioni, ovvero le informazioni che l'utente sta cercando.

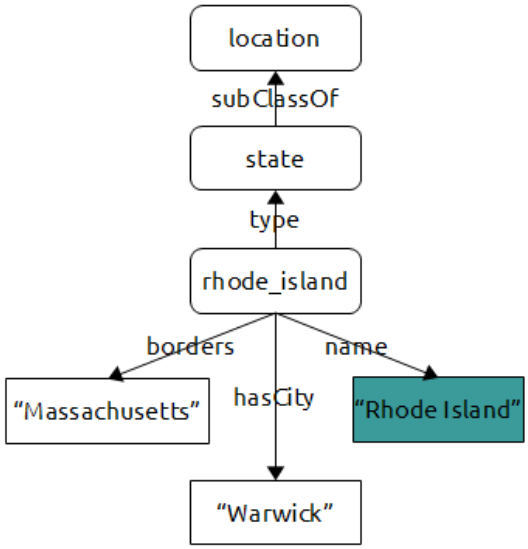


Figura 13: porzione di geobase che soddisfa la query 1

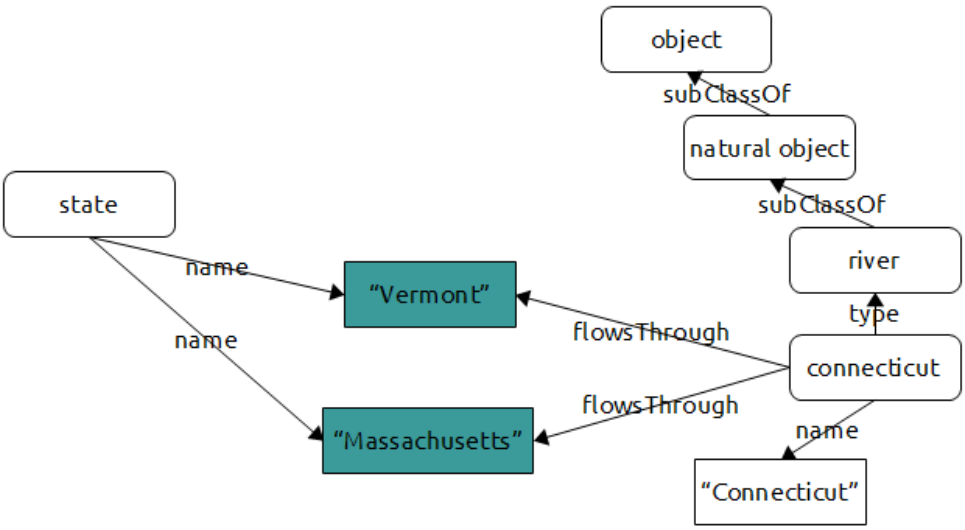


Figura 14: porzione di geobase che soddisfa la query 2

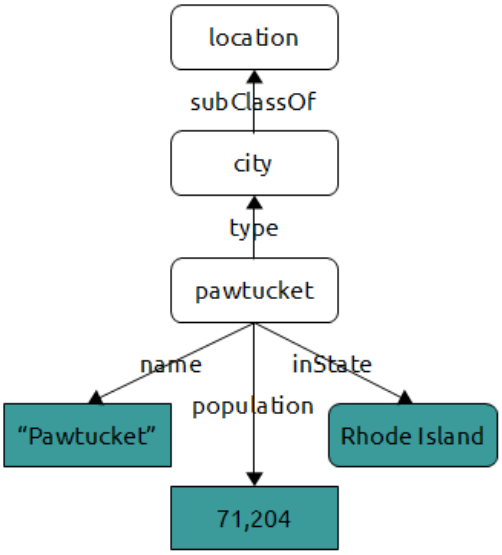


Figura 15: porzione di geobase che soddisfa la query 3

Come si puo' notare questi grafi contengono tutti i dettagli relativi allo schema dei dati: le gerarchie sono completamente specificate, gli attributi sono invocati tramite i termini specifici presenti nel vocabolario; risulta evidente che la distanza dal linguaggio naturale e' molto pronunciata.

GeX e' uno strumento che mette l'utente nella condizione di poter generare query flessibili ed approssimative; per fare questo fornisce, come gia' accennato, un linguaggio semanticamente ricco per la definizione di query basate su struttura a grafo: le interrogazioni sono viste come multigrafi che connettono nodi entita' a nodi variabile (che contengono i valori delle entita') e dove le condizioni della query sono specificate nelle condizioni poste sui nodi variabile; tale linguaggio fornisce all'utente diversi gradi di flessibilita', ad esempio consente l'utilizzo sia di collegamenti diretti (orientai) che di collegamenti indiretti (non orientati) e

permette l'uso di variabili come etichette degli archi, rendendo cosi' possibile l'instaurarsi di relazioni tra nodi non associati direttamente.

Riassumendo, cio' che GeX permette agli utenti di fare e' di interrogare un dataset tramite una query piu' vicina a quello che e' il loro linguaggio naturale piuttosto che al vocabolario fissato dello schema dei dati.

Per chiarire meglio le funzionalita' del software proponiamo la rappresentazione grafica delle traduzione delle query nel linguaggio fornito da GeX; sottolineiamo che, data l'elevata flessibilita' del linguaggio, una stessa query puo' essere tradotta in diversi modi e quindi avere diverse rappresentazioni grafiche; qui si presentano solo alcuni esempi.

Per agevolare nella lettura dei grafi facciamo presente che i nodi e gli archi etichettati con label del tipo "*$lettera*" (dette *any label*) rappresentano nodi o archi il cui contenuto non e' specificato.

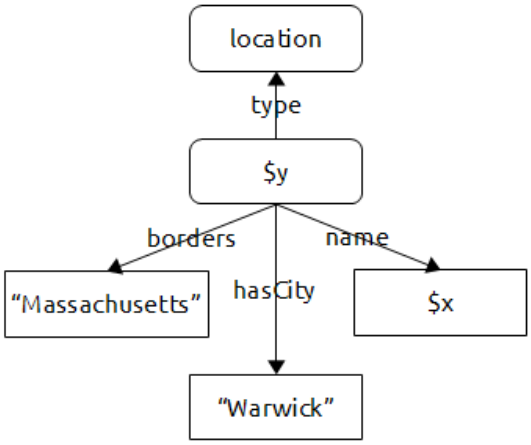


Figura 16: grafo di una delle possibili forme della query 1

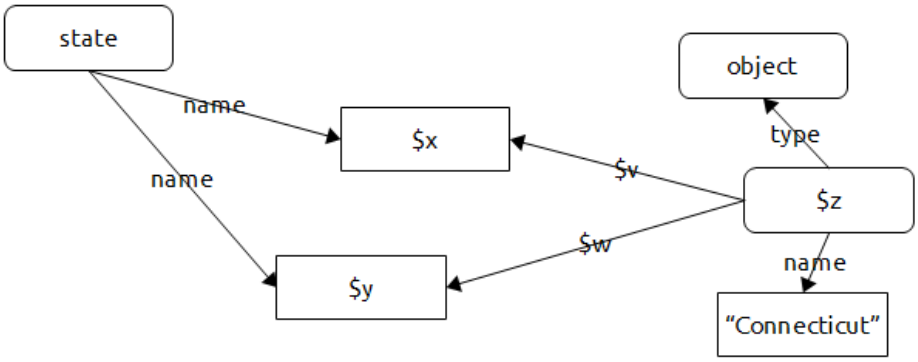


Figura 17: grafo di una delle possibili forme della query 2

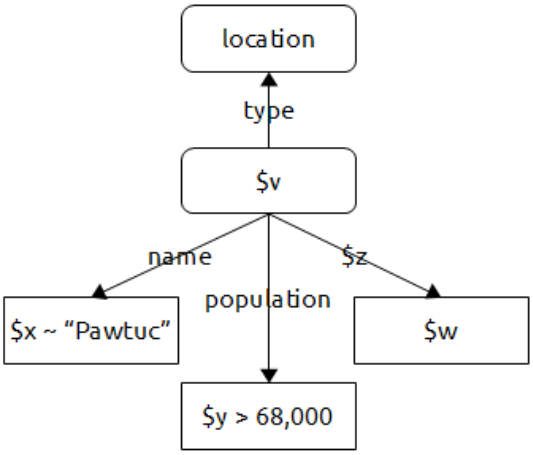


Figura 18: grafo di una delle possibili forme della query 3

Se traduciamo il grafo in linguaggio naturale otteniamo frasi molto simili alle query in linguaggio naturale che stiamo esaminando, per esempio il grafo in figura 14 puo' essere tradotto in "Gli stati di cui non conosco il nome che sono connessi in un qualche modo ad un oggetto che ha nome "Connecticut"."

Date queste query, il meccanismo di ricerca di corrispondenze si basa su un algoritmo di recupero top-k che restituisce i sottografi del grafo completo (per esempio le porzioni rappresentate in figura 10, 11 e 12) che

meglio corrispondono alle necessita' dell'utente. A tal fine GeX implementa un approccio di corrispondenza approssimativa di un sottografo che rispetta i vincoli specificati nella query implementando un meccanismo che consente di avere ambiguita' sia nelle label che etichettano i nodi, sia nelle relazioni tra questi. Concretamente questo aspetto e' visibile nella figura 15, dove abbiamo nodi e relazioni etichettati con any label.

Questo meccanismo di corrispondenza approssimativa e' implementato da due funzionalita' fondamentali del software: l'*approssimazione della struttura* e l'*approssimazione delle label che etichettano nodi e archi*.

Per quanto riguarda il primo punto il software persegue l'obiettivo di estrarre solo le approssimazioni di struttura piu' significative, ovvero GeX trova corrispondenze di relazioni tra due nodi di una query nel dataset solo se questi nodi sono semanticamente connessi nel dataset stesso. Concretamente, osservando la figura 10, notiamo che l'oggetto con nome "*Rhode Island*" e' un'istanza che deriva della classe *location* e per questo rispetta i vincoli imposti dalla query 1, il software quindi estrae quel percorso in quanto le due entita' sono semanticamente connesse e rispettano i vincoli imposti dalla query.

Una giusta osservazione riguarda il fatto che in un grafo i percorsi che si possono contare tra i nodi sono potenzialmente moltissimi, proporzionali al numero di nodi, e non tutti questi percorsi sono concretamente utili per rispondere ad una query; cio' complica in buona misura il lavoro di estrazione dei dati. Per risolvere tale problema GeX si basa su un modello di indicizzazione dei percorsi chiamato *SR-index* (indicizzazione *Semantic Relatedness*) modello che ci permette di controllare in modo efficiente se due nodi sono semanticamente connessi o no sfruttando i vincoli imposti dalle label specificate nelle query. Cio' restringe significativamente

l'insieme di percorsi che si devono analizzare e che corrispondono alle richieste dell'utente.

Per chiarire ancora meglio quello che GeX e' in grado di fare osserviamo la figura sotto:

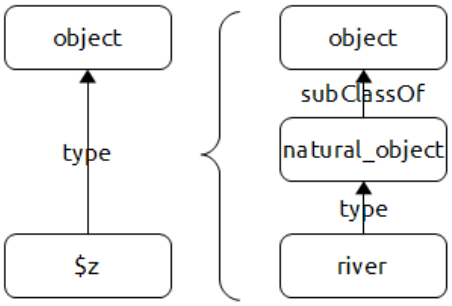


Figura 19: dettaglio del processo di approssimazione della struttura della query 2

A sinistra abbiamo la coppia di nodi rappresentata nella query 2, figura 14, mentre a destra abbiamo il percorso che GeX rintraccia ed estrae dal dataset geobase. Come si vede il software e' in grado di "esplodere" i percorsi che vengono sottintesi e fusi all'interno delle query estraendo i percorsi effettivi che si trovano nel dataset sulla base di quelle che sono le specifiche delle istanze (qui non riportate ma parliamo del valore del campo "*name*" = "*Connecticut*").

Come si nota dalle figura 17 riportata sotto, GeX non solo permette la mancanza di informazioni per label che rappresentano entita' (come il nome di uno stato) ma permette l'uso di any label anche su archi, ovvero permette all'utente di esprimere informazioni parziali del tipo "so che due entita' sono connesse, ma non conosco l'effettiva natura di tale connessione". E' quello che accade nel grafo in figura 14 e sotto riportiamo un dettaglio della query messo in relazione con il percorso che viene estratto dal dataset di geobase:

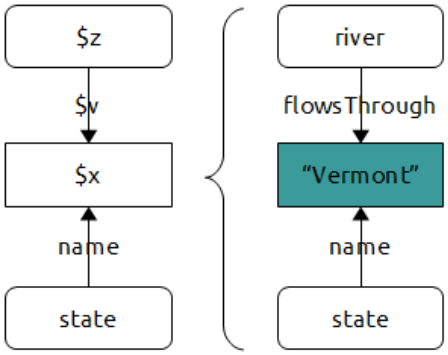
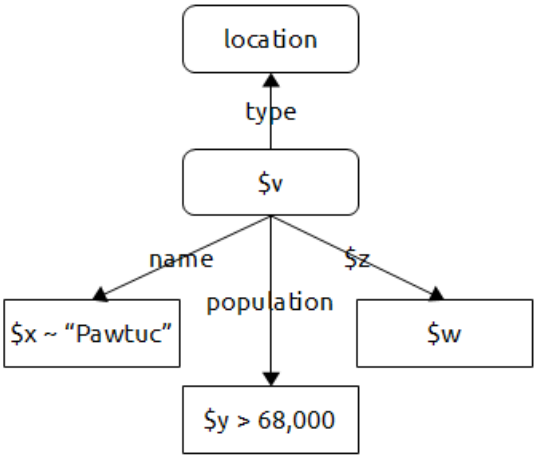


Figura 20: dettaglio del processo di approssimazione della struttura della query 2

Il secondo tipo di approssimazione riguarda le etichette dei nodi e degli archi.

Come sappiamo il linguaggio naturale e' estremamente vasto per cui e' molto difficile che gli utenti utilizzino gli stessi termini che appartengono al vocabolario dei dati; inoltre puo' verificarsi che non sappiano indicare in modo preciso il nome di un'istanza; l'esempio in figura 15 (riportata anche sotto per semplicita') mostra l'interrogazione di un utente di estrarre i dati dei luoghi che hanno un nome simile a "Pawtuc" e una popolazione maggiore di 68.000 abitanti.



Per affrontare queste problematiche GeX e' predisposto con un meccanismo di riconoscimento della similarita' delle label, ovvero in un

contesto in cui alcune label non corrispondono alle label presenti nel set dei dati il software calcola la distanza che intercorre tra la label proposta dall'utente e quelle presenti nel dataset e, di queste, tiene in considerazione le label che hanno una "distanza" minore di un certo limite prefissato. Questo limite ci dice che al di sotto di esso (per valori piu' piccoli) la label in esame si avvicina a quello che l'utente ha richiesto, per valori piu' grandi significa che siamo troppo lontani dalle richieste dell'utente e quindi dobbiamo scartare l'etichetta.

Concludendo possiamo affermare che le funzionalita' implementate dal software in esame rendono possibile l'interrogazione approssimata e flessibile di dati modellati a grafo; andiamo ora ad esaminare gli strumenti utilizzati per trasformare le interrogazioni dal linguaggio naturale ad una struttura formale basata su una rappresentazione a grafo.

2.3. Analisi linguistica di frasi in linguaggio naturale - C&C e Boxer

Boxer e' un software sviluppato da Johan Bos per l'analisi linguistica di testi e lavora insieme al parser C&C sviluppato da James Curran e Stephen Clark [6]. I software sono rilasciati insieme sotto licenza non commerciale e sono entrambi sono scritti in C++ e il loro obiettivo e' quello di eseguire l'analisi morfologica di frasi in linguaggio naturale per restituirne poi una rappresentazione formale.

I due software lavorano in tempi diversi: inizialmente le frasi prese in esame vengono parsate e analizzate da C&C, successivamente Boxer genera una rappresentazione grammaticale formale (in vari formati e linguaggi, a seconda delle richieste dell'utente) dei dati ricevuti da C&C.

Analizziamo in breve il lavoro eseguito dal parser.

C&C [7] e' uno strumento costruito sulla base del CCG (*Combinatory Categorial Grammar*), un formalismo grammaticale efficientemente parsabile e linguisticamente espressivo: tale grammatica rappresenta un'interfaccia trasparente tra la superficie sintattica e la sottostante rappresentazione semantica dei termini che si trovano in una frase; CCG e' un formalismo grammaticale lessicalizzato, ovvero ogni parola di una frase viene esaminata ed assegnata ad una struttura sintattica elementare tra quelle presenti nella CCGbank, archivio dove sono contenute un gran numero di categorie lessicali, regole attraverso cui correlare queste categorie e cosi' via. Utilizzare questo tipo di grammatica CCG risulta molto vantaggioso in quanto si basa su poche regole grammaticali e si fonda sul principio di *type-trasparency* secondo cui ogni tipo sintattico corrisponde ad uno e un solo tipo semantico.

L'obiettivo di C&C e' quindi quello di parsare secondo queste regole le frasi che gli vengono fornite in input, e generare un output che successivamente verra' poi analizzato.

Boxer [8] ha il compito di produrre una rappresentazione formale linguistica della struttura in output dal parser CCG. Per fare questo non

solo sfrutta la teoria della grammatica CCG, ma si basa anche sulla teoria DRT (*Discourse Representation Theory*). Quest'ultima teoria e' uno strumento che offre un linguaggio ricco per la rappresentazione del significato contestuale che si trova all'interno di un discorso, permettendo cosi' non un'analisi grammaticale asettica ma un'analisi che dipende maggiormente dal contesto e dalla semantica della frase. Per fare questo Boxer si avvale, oltre che delle categorie della grammatica CCG, anche di una componente di C&C, il POS (*Part Of Speech*) tagger che accede alla radice morfologica di una parola.

Una volta analizzato il risultato del parsing di C&C, Boxer puo' generare output in vari formati (prolog, latex o XML) e secondo diverse sintassi (DRS, RDF, CCG ecc.) a seconda dell'uso che se ne deve fare.

Per l'ambito di questa tesi si e' scelto un output basato sulla semantica RDF in formato XML: queste opzioni generano un output strutturalmente simile al formato delle query che sono utilizzate da GeX per l'interrogazione dei dati.

Di seguito riportiamo il risultato delle operazioni di C&C e Boxer sulle prime due query presentate precedentemente; precisiamo che Boxer e C&C sono software che lavorano in lingua inglese, quindi le interrogazioni sono state tradotte in tale lingua come segue:

Query 1: "*The locations bordering Massachusetts that have a city named Warwick.*"

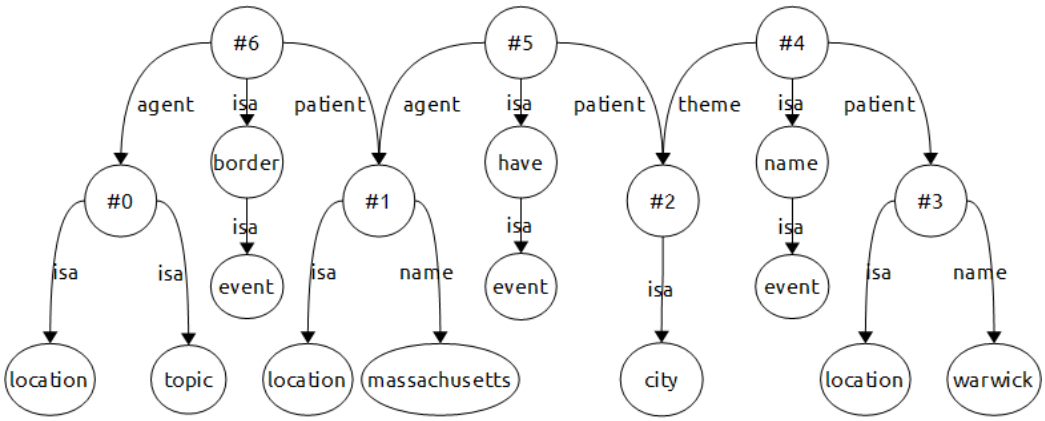


Figura 21: grafo relativo all'analisi grammaticale della query 1

Query 2: "*The states related to an object named Connecticut.*"

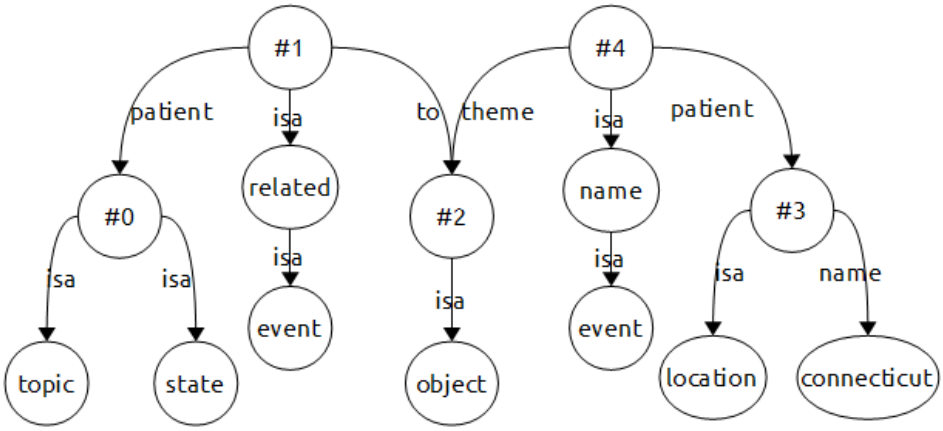


Figura 22: grafo relativo all'analisi grammaticale della query 2

L'analisi grammaticale di una frase dipende strettamente da come la frase e' costruita; ne deriva che, data l'elevata ambiguita' del linguaggio naturale, uno stesso concetto puo' essere espresso in moltissime forme diverse, ottenendo cosi' diverse analisi grammaticali e diversi grafi.

Parte II

Tecniche per l'interrogazione in linguaggio naturale di dati modellati a grafo.

Capitolo 3

Studio delle trasformazioni

In questo capitolo presenta il lavoro svolto concretamente per lo sviluppo della tesi. Verra' esposto lo scopo del lavoro, le assunzioni che si sono fatte e tutto il percorso di ricerca di pattern ricorrenti.

3.1. Scopo finale

L'obiettivo che ci siamo prefissati con questa tesi e' stato quello di rintracciare delle porzioni di grafi, dette *pattern*, presenti nei grafi grammaticali di Boxer, che si presentassero con una certa frequenza nei grafi in output, indipendentemente dalla frase sottoposta al parser; oltre a ritracciare i pattern piu' diffusi si sono cercate le loro possibili trasformazioni in modo da generare dei grafi piu' vicini allo schema dei dati e alla struttura delle query utilizzate da GeX.

Il lavoro di ricerca dei pattern e' stato svolto su una cinquantina di interrogazioni diverse.

3.2. Alcune note e assunzioni

Di seguito riportiamo le assunzioni che si sono fatte nel corso del lavoro e alcune note sul comportamento dei software.

Per quanto riguarda Boxer ogni volta che si inserisce il nome di un luogo, di un monte, di una persona in un'interrogazione lo si deve scrivere con la lettera maiuscola; in caso contrario il grafo lessicale risultate e' molto diverso da quello corretto.

Le interrogazioni (sia domande che affermazioni) devono essere poste nel seguente formato:

Le citta' nello stato del Maine .

La punteggiatura dev'essere separata da uno spazio dall'ultima parola; e' sconsigliato l'uso di abbreviazioni tipo "*What's*" in quanto il parser non riconosce la stringa; si consiglia di iniziare le frasi con un articolo, per esempio *The*, *A*, *An*, o una tra le parole interrogative quali *What*, *Where*, *Which*, *How*; in ogni caso e' consigliato l'uso della maiuscola.

La punteggiatura interna alla frase o i caratteri come i doppi apici (") non vengono interpretati ma semplicemente ignorati e mantenuti annessi alla parola a cui sono vicini. Se si vogliono parsare piu' frasi nello stesso momento basta dividerle con un carattere *a capo* e Boxer generera' un grafo per ogni frase. Non e' necessario che le frasi siano correlate tra loro, il parser le analizza separatamente senza tenere conto del contesto globale del file contenente le interrogazioni.

Per essere considerato valido l'output di Boxer deve generare un grafo in cui i nodi sono connessi, ovvero non ci devono essere uno o piu' nodi distaccati.

Boxer non e' in grado di parsare frasi contenenti numeri o codici numerici: non genera alcun grafo in output. L'unico modo per inserire un numero in un'interrogazione e' quello di scriverlo racchiudendolo tra doppi apici ("), ma non sempre il parser lavora bene.

Nel caso si parsino frasi in cui si fa riferimento a un valore di una determinata propieta', ad esempio l'abbreviazione "AK" per uno stato, si consiglia di racchiuderla tra doppi apici (") ovvero di scrivere una frase del tipo

The rivers with named "Red".

Questo formalismo permette di riconoscere alcuni pattern particolari. In questo caso, come nel caso dei numeri tra doppi apici, nel momento in cui il grafo di Boxer e' trasformato in uno utile per GeX alle label in cui il carattere iniziale e finale corrispondono al carattere doppi apici (") verra' concatenata una stringa del tipo *^^xsd:string* o *^^xsd:unsignedInt*; tale stringa e' necessaria a GeX per trovare corrispondenze tra la label dei nodi della query e quelle memorizzate nei database: tutti i nodi valore devono quindi essere nella forma *"stringa"^^xsd:string* nel caso delle stringhe, *"numero"^^xsd:unsignedInt* nel caso di valori numerici. Per chiarezza nei grafi in esempio tali apici non saranno riportati fatta eccezione per il caso specifico del pattern che si interessa di tale struttura.

Nel caso si debbano parsare stringhe di piu' parole che rappresentano il nome di un'entita', ad esempio di una citta' o il titolo di un libro, e' bene sostituire gli spazi tra le parole con un carattere speciale che non sia uno tra {#, $, -, _, &} o la punteggiatura classica; per esempio si potrebbe usare il carattere % ed ecco due esempi:

"New York" -> "New%York"

"Non-Deterministic Two-Tape Automata are More Powerful Then Deterministic Ones." -> "Non-Deterministic%Two-Tape%Automata%are%More%Powerful%Then%Deterministic%Ones."

L'inserimento di questo carattere speciale evita che Boxer interpreti tutte le parole del nome dell'entita' come inerenti alla frase che si sta parsando, generando quindi un grafo non coerente con le nostre richieste.

Inoltre il secondo esempio ci mostra come mai e' sconsigliato usare il carattere -, dato che e' gia' presente nel nome dell'oggetto. Nel momento in cui il grafo viene trasformato i caratteri speciali verranno rimossi dato che GeX deve ricercare corrispondenze con il nome effettivo e reale dell'instanza.

Da questo momento in avanti non si definiranno piu' le any label tramite le stringhe "*$lettera*" ma tramite "*#numero*": il significato delle due notazioni e' analogo e non genera problemi da parte di GeX nell'interrogazione dei dati.

Su di un grafo completamente trasformato e pronto per essere utilizzato da GeX va eseguita una trasformazione per tutte le label degli archi del tipo "*isa*" in "*type*"; GeX infatti interpreta gli archi *isa* in maniera differente dal significato che essi hanno in Boxer, verrebbero quindi a crearsi degli errori anche per query corrette. Nel caso si debbano effettuare modifiche su altre label queste verranno riportate nei vari casi specifici.

3.3. Studio e ricerca dei pattern

In questo paragrafo vengono presentati tutti i pattern che sono stati rintracciati durante la fase di sperimentazione.

Per ogni pattern sono riportati esempi di query in inglese (con relativa traduzione in italiano), il corrispondente grafo generato da Boxer e il grafo trasformato grazie ai pattern. Per ogni esempio verranno riportati i pattern applicati e su quali nodi. Nota importante: l'analisi del grafo, e quindi l'applicazione dei pattern, viene sempre svolta a in ordine crescente, dal nodo con indice del tipo *#numero* minore a quello con indice maggiore.

In questa fase di sviluppo della tesi si e' lavorato osservando esclusivamente lo schema di dati RDF geobase presentato precedentemente (capitolo 2, paragrafo 2.1) e gli sono state sottoposte sia interrogazioni in forma di affermazione, sia in forma interrogativa.

Per aiutare nella lettura dei paragrafi che seguono inseriamo una tabella riassuntiva con query, pattern applicati e riferimenti ad essi.

n. query	Query	Pattern applicati
1	"What are the states?"	1, 2
2	"What are the cities?"	1, 2
3	"The capital of Iowa."	3, 4
4	"What is "Flathead"?"	1, 2, 4
5	"The entities named Colorado."	4, 5
6	"What are the neighboring states for Michigan?"	1, 2, 4, 6
7	"The height of the Mount Sunflower."	3, 4, 7
8	"What is the population of Boulder?"	1, 2, 4, 7
9	"The cities in Alaska."	3, 4, 8
10	"What are the high points of the states surrounding Alabama?"	1, 2, 4, 7, 9, 10
11	"The states with a city named "Akron"."	3, 4, 11, 16, 19

12	"The states that have a city named "Akron"."	3, 4, 10, 11
13	"What are the cities which are in California?"	1, 2, 4, 12
14	"What is the number of people in Dallas?"	1, 2, 4, 13
15	"The lowest point in Illinois."	3, 4, 8, 14
16	"The states related to an object named "Centerville"."	3, 4, 5, 11, 15, 16
17	"What are the states that the Potomac runs through?"	1, 2, 4, 17
18	"The state abbreviated "ar"."	10, 18
19	"What is the capital city in Florida?"	1, 2, 4, 19
20	"The locations border Nebraska and that have a city named "Casper"."	4, 10, 11, 16, 19
21	"How high is mount Mckinley?"	-
22	"Where is Indianapolis?"	-
23	"Which lakes are in Minnesota?"	2, 4, 12
24	"Which capitals are in the states that border Nebraska?"	-
25	"Which states is Kalamazoo in?"	-

Pattern n. 1 e n. 2

Vediamo alcune frasi e i rispettivi grafi derivanti dal parsing con Boxer.

Query1: "*Quali sono gli stati?*" -> "*What are the states?*"

Query 2: "*Quali sono le citta'?*" -> "*What are the cities?*"

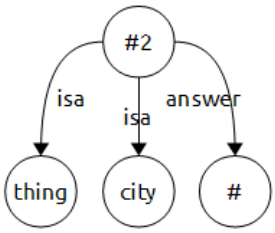


Figura 23: grafo del parsing della query 1

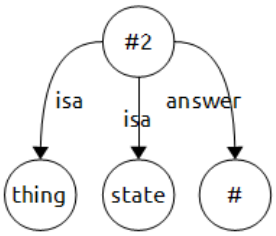


Figura 24: grafo del parsing della query 2

In questi grafi abbiamo due rami che non sono necessari nelle query da sottoporre a GeX e nello specifico ci riferiamo agli percorsi (*#2, isa, thing*) e (*#2, answer, #*).

Questi percorsi sono rintracciabili in tutte le interrogazioni che terminano con "?" e quindi in tutte le domande.

La loro semplificazione prevede l'eliminazione completa dell'arco e del nodo di arrivo:

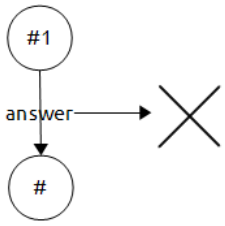


Figura 25: pattern 1

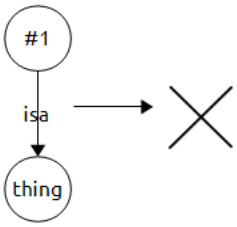


Figura 26: pattern 2

I nostri esempi verranno quindi semplificati come segue:

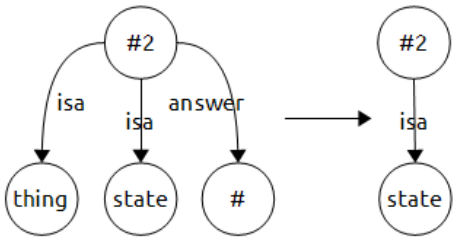


Figura 27: trasformazione grafo query 1

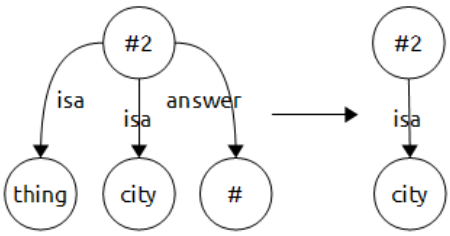


Figura 28: trasformazione grafo query 2

I grafi risultati possono essere sottoposti a GeX e restituiranno rispettivamente l'elenco degli stati d'America e delle citta'.

Pattern n. 3

Query 3: "*La capitale dell'Iowa.*" -> "*The capital of Iowa.*"

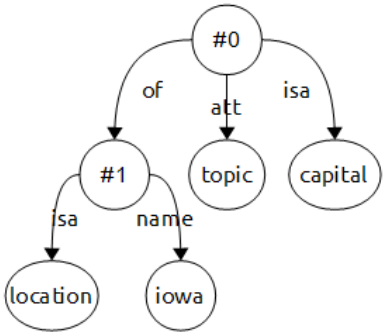


Figura 29: grafo del parsing della query 3

Grazie a questo esempio possiamo rintracciare un pattern molto semplice che si trova in tutte le interrogazioni che terminano con ".": si tratta del percorso (*#0, att, topic*) che, come i precedenti due pattern, non e' utile al fine delle interrogazioni tramite GeX e per questo puo' essere completamente eliminato.

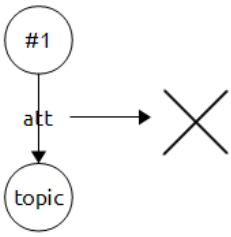


Figura 30: pattern 3

Il nostro esempio viene quindi trasformato come segue:

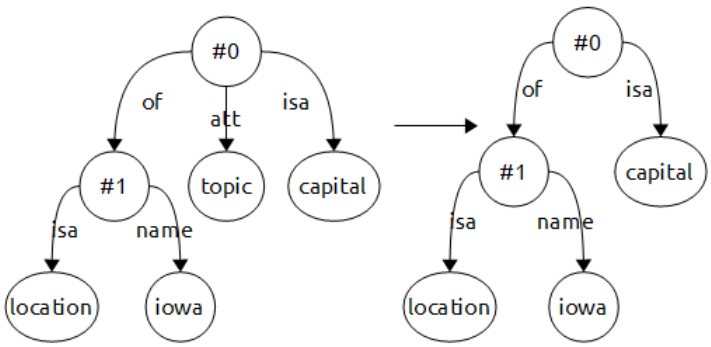


Figura 31: semplificazione grafo query 3

Il grafo ottenuto deve essere ulteriormente trasformato.

Pattern n. 4

L'esempio precedente ci torna utile per esporre il prossimo pattern. Facendo riferimento alla figura 28 osserviamo questo particolare:

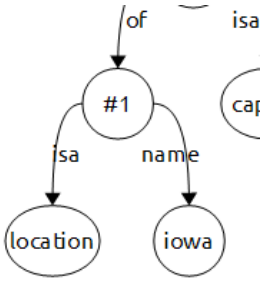


Figura 32: particolare della figura 28 riferita al pattern 4

Questo pattern si rintraccia tutte le volte che in una query si inserisce un nome proprio di persona, di uno stato, di un fiume e cosi' via, ovvero ogni volta che all'interno dell'interrogazione in linguaggio naturale inseriamo una o piu' parole che iniziano con la lettera maiuscola. Il nodo che qui e' etichettato con "*location*" puo' avere anche etichette del tipo "*entity*", "*organization*", "*person*" e il suo compito e' quello di specificare a che categoria appartiene l'oggetto #1 che ha nome "*iowa*". Queste categorie non sempre corrispondono all'effettiva natura dell'oggetto #1

che l'utente intende, ne abbiamo un esempio nella figura sotto, dove l'entita' con nome "*flathead*" non e' un'organizzazione ma bensi' un lago del Montana.

Query 4: "*Che cos'e' "Flathead"?*" -> "*What is "Flathead"?*"

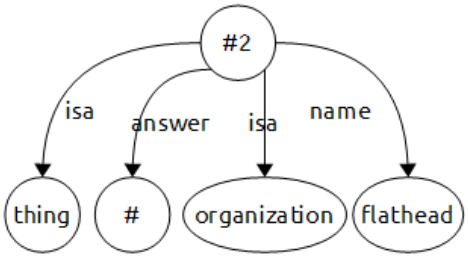


Figura 33: grafo del parsing della query 4

Considerando la flessibilita' offertaci da GeX possiamo notare che un attributo come "*location*" non e' strettamente necessario per la ricerca di corrispondenze, infatti il software, senza la specificazione di tale attributo, restituisce tutte le entita' che hanno nome "*flathead*". Inoltre, osservando lo schema dei dati, notiamo che questo tipo di attributi non si trova nello schema (nel caso specifico di geobase abbiamo una classe del tipo "oggetto" ma e' una coincidenza); per questo se lasciamo l'attributo e' probabile che saranno maggiori le volte in cui non vengono trovate corrispondeze, piuttosto che le volte in cui ottengo risultati. Ne deriva che possiamo eliminare l'arco e il nodo di destinazione in tutte le sue forme.

Sfruttiamo la presenza di questi attributi derivanti dall'analisi grammaticale di Boxer per appendere alla stringa all'attributo *name* una stringa del tipo *"*nome_oggetto*"^^xsd:string*; si tratta di una stringa utile a GeX per riconoscere degli attributi *name* all'interno dello schema dei dati. Senza queste stringhe il software non da' luogo a corrispondenze con le label salvate nella base di dati.

Ulteriore operazione che si puo' compiere e' quella di controllare la label connessa tramite l'arco *name*: se la label contiene il carattere speciale "_" (underscore) lo si deve sostituire con uno spazio; questo processo e'

necessario per le entita' con nome composto come "New York" o "Oceano Atlantico". Boxer infatti riconosce questi nomi come nomi di location e li parsa concatenandoli con un underscore; tale formattazione pero' non corrisponde a quella presente nei dataset, dove gli attributi *name* possono contenere spazi o altro. E' quindi utile fare anche questo controllo e, nel caso, procedere con la modifica della stringa.

Mostriamo quindi il pattern numero 4:

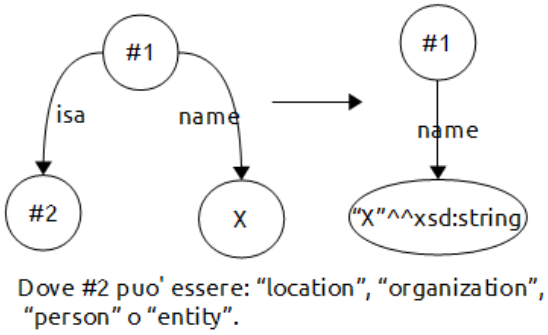


Figura 34: pattern 4

E ora i risultati dell'applicazione dei pattern visti sulle query 3 e 4 in esempio:

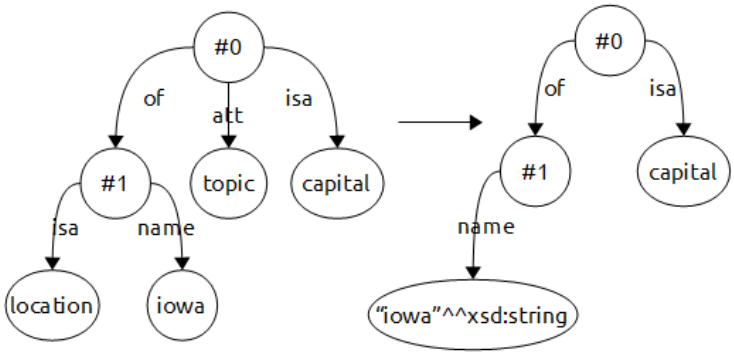


Figura 35: processo di semplificazione della query 3

Come si vede abbiamo applicato anche il pattern 3 insieme con il 4.

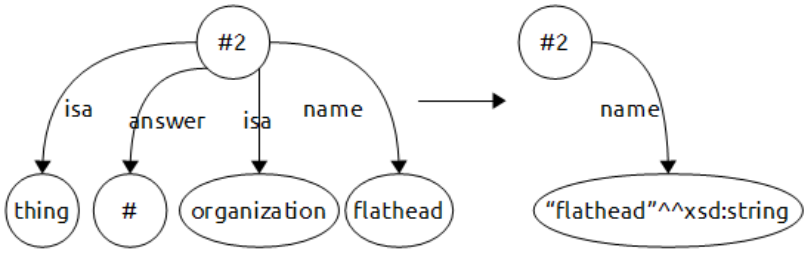


Figura 36: processo di semplificazione della query 4

Insieme con il pattern 4 abbiamo applicato anche l'1 e il 2.

I grafi ottenuti possono essere usati da GeX e daranno risultati.

Per semplicita', da questo momento in poi non appenderemo piu' la stringa completa, ma inseriremo l'attributo *name* tra doppi apici (").

Pattern n. 5

Esaminiamo un pattern simile a quello precedente. Consideriamo la seguente query:

Query 5: "*Le entita' che si chiamano "Colorado".*" -> "*The entities named "Colorado".*"

Otteniamo il seguente grafo ne evidenziamo un dettaglio.

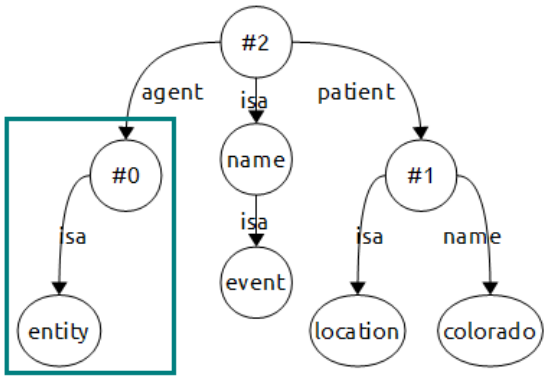


Figura 37: grafo del parsing della query 5

Il percorso evidenziato ci dice che stiamo cercando un'istanza di tipo "*entity*". Notiamo che siamo in un caso differente da quello evidenziato al passo precedente: nel pattern 4 la specifica del tipo di oggetto a cui appartiene una certa entita' viene generata automaticamente dal parser, ed e' quello che accade anche in questo esempio al nodo #1; nel particolare evidenziato invece *entity* deriva da una nostra personale specifica nella formulazione della query: vogliamo un'entita', ovvero vogliamo un qualsiasi oggetto con determinate specifiche.

Il nodo che consideriamo puo' contenere sia la label *entity* che *object* nel caso in cui la mia query sia del tipo "*Gli oggetti chiamati...*".

Il pattern in esame puo' avere due possibili risoluzioni. La prima, come nel caso precedente, prevede l'eliminazione dell'arco e del nodo di destinazion; la seconda invece e' una risoluzione piu' fine e prevede la trasformazione della label del nodo di destinazione in una any label:

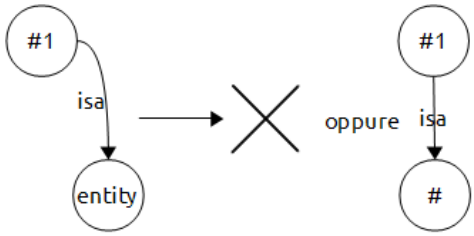


Figura 38: pattern 5

Abbiamo sostituito la label *entity* con la label *#*, ovvero in linguaggio naturale potremmo tradurlo come "*Tutte le istanze che sono di un qualsiasi tipo*".

Le due trasformazioni proposte, applicate al grafo della query 5, generano i seguenti risultati:

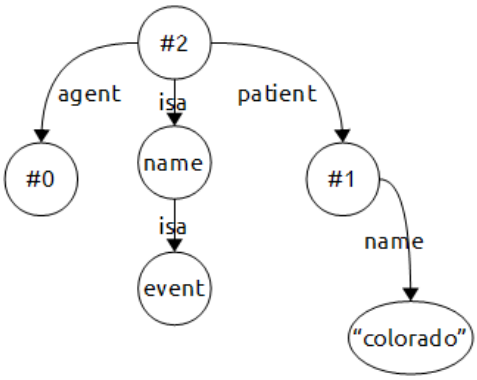


Figura 39: possibile semplificazione del grafo della query 5

Oppure:

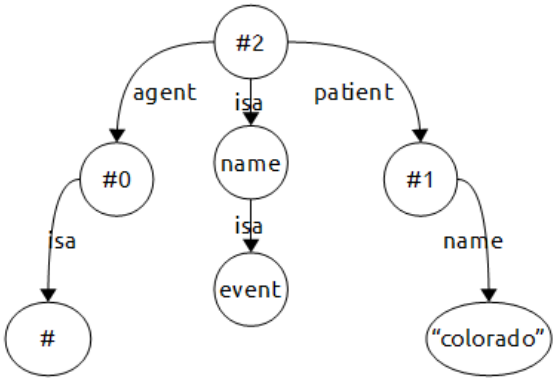


Figura 40: possibile semplificazione del grafo della query 5

Entrambi i grafi necessitano di altre trasformazioni.

Pattern n. 6

In diverse query e' stato rintracciato il pattern che vi andiamo ad illustrare con la relativa semplificazione:

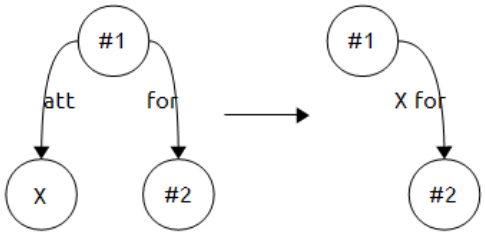


Figura 41: pattern 6

Il pattern e' sfruttato per avvicinare i grafi grammaticali di Boxer nei grafi usati da GeX: spostando un attributo su un arco infatti ci allontaniamo dal formalismo grammaticale per avvicinarci al vocabolario dello schema dei dati. Non importa se i termini non corrispondono strettamente al vocabolario, in un secondo momento potra' essere previsto un meccanismo di ricerca di label simili a quella presente nella query.

Guardiamo un esempio concreto:

Query 6: "*Quali sono gli stati vicini del Michigan?*" -> "*What are the neighboring states for Michigan?*"

Ed ecco il risultato del parsing di C&C e Boxer:

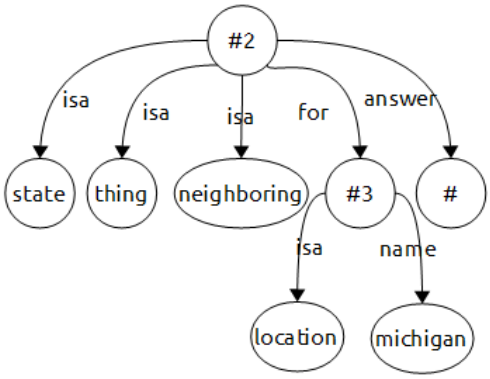


Figura 42: grafo del parsing della query 6

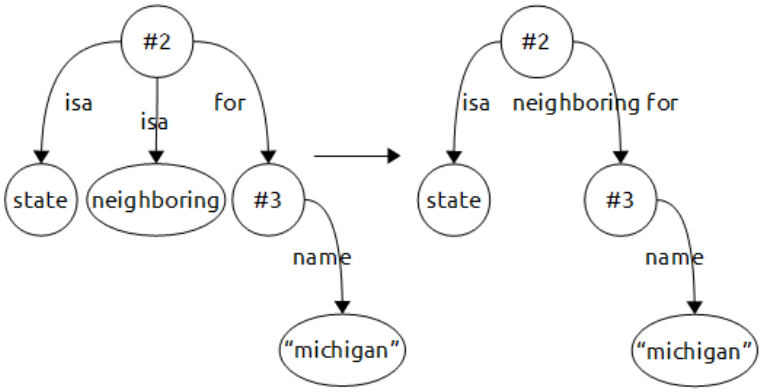


Figura 43: trasformazioni grafo query 6

In figura 40 abbiamo i passaggi delle trasformazioni del grafo. Abbiamo applicato i pattern 1 e 2 al nodo #2 e il pattern 4 al nodo #3, successivamente e' stato usato il pattern 6 sul nodo #2. Il grafo generato puo' essere usato da GeX previa modifica della label "*neighboring for*" in "*borders*".

Con questo esempio ci troviamo ad evidenziare un problema che puo' essere riscontrato in vari casi, ovvero quello dei conflitti derivanti dall'avere piu' percorsi che potrebbero essere utilizzati nella trasformazione; in questo caso l'arco *for* potrebbe essere trasformato ininsieme con il percorso (*#1, isa, state*) o con (*#1, isa, neighboring*).

Quale arco dobbiamo scegliere? Se si pensa ad un'applicazione delle trasformazioni a livello automatico una soluzione puo' essere quella di controllare piu' approfonditamente il contenuto dei nodi tra i quali si puo' decidere: in questo caso, infatti, il nodo *state* indica la tipologia di entita' che si sta ricercando ed e' un termine legato allo schema dei dati; il nodo *neighboring*, invece, e' etichettato con un termine che non e' presente nello schema ed e' quindi probabile che si riferisca ad un attributo o ad una propieta' (nello specifico lo si puo' tradurre con *borders*), ed e' per questo un candidato ideale per la trasformazione.

Pattern n. 7

Il pattern che mostriamo ora e' un pattern diffuso nelle interrogazioni in cui compare la parola "*of*". Il grafo che si ottiene dal parsing della frase presenta sempre struttura specifica:

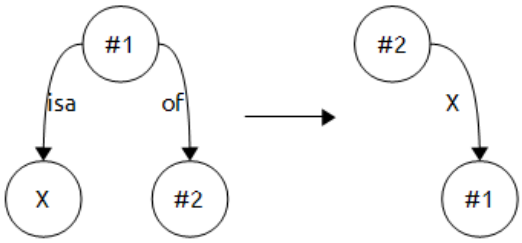


Figura 44: pattern 7

Vediamo alcuni esempi:

Query 7: "*L'altezza del Monte Sunflower.*" -> "*The height of the Mount Sunflower.*"

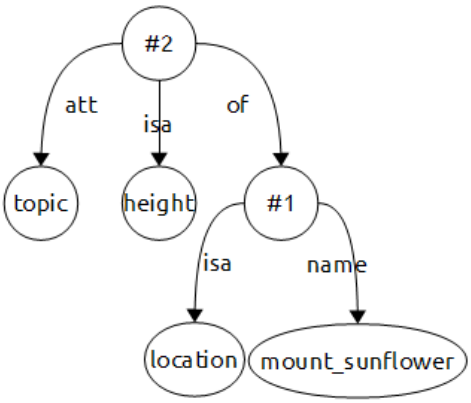


Figura 45: grafo del parsing della query 7

Notiamo immediatamente che possiamo applicare i pattern 3 e 4 al nodo #2 e #1, ottenendo una versione semplificata del grafo; inoltre, come mostra la figura 42, applichiamo anche il pattern 7.

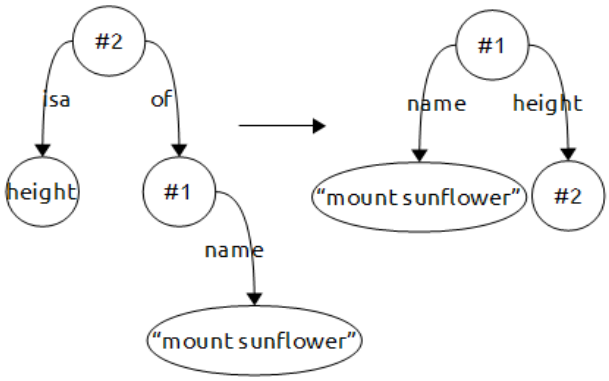


Figura 46: trasformazione del grafo della query 7

Il grafo ottenuto puo' essere utilizzato da GeX.

Query 8: "*Qual e' la popolazione di Boulder?*" -> "*What is the population of Boulder?*"

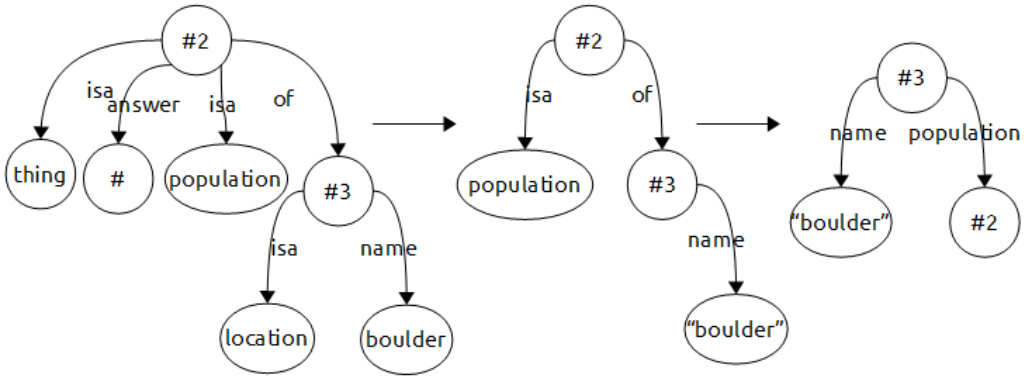


Figura 47: processo di trasformazione della query 8

Nel dettaglio abbiamo applicato i pattern 1 e 2 al nodo #2, il pattern 4 al nodo #3; successivamente abbiamo usato il pattern 7 sul nodo #2. Il grafo puo' essere cosi' utilizzato da GeX.

Pattern n. 8

Questo pattern e' molto simile al precedente con la differenza che si presenta nelle frasi in cui utilizziamo il termine "*in*". Di seguito la struttura del pattern e un esempio di trasformazioni di query.

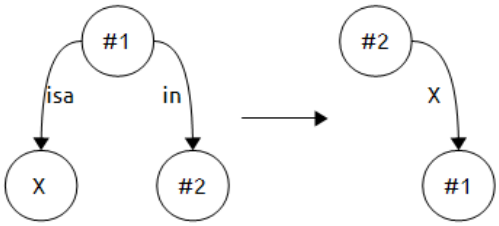


Figura 48: pattern 8

Query 9: "*Le citta' in Alaska*" -> "*The cities in Alaska.*"

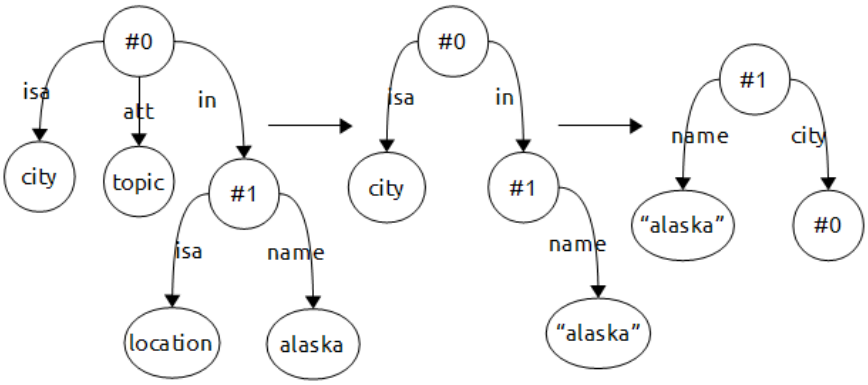


Figura 49: processo di trasformazione della query 9

In questo caso abbiamo applicato il pattern #3 al nodo #0 e il pattern 4 al nodo #2; successivamente abbiamo usato il pattern 8 sul nodo #2. Il grafo puo' essere utilizzato da GeX previa modifica della label "*city*" in "*hasCity*"

Pattern n. 9

Presentiamo ora una query piu' complessa che ci permettera' di lavorare non solo sul pattern 9 ma anche sul 10.

Query 10: "*Quali sono i punti piu' alti degli stati che confinano con l'Alabama?*" -> "*What are the high points of the states surrounding Alabama?*"

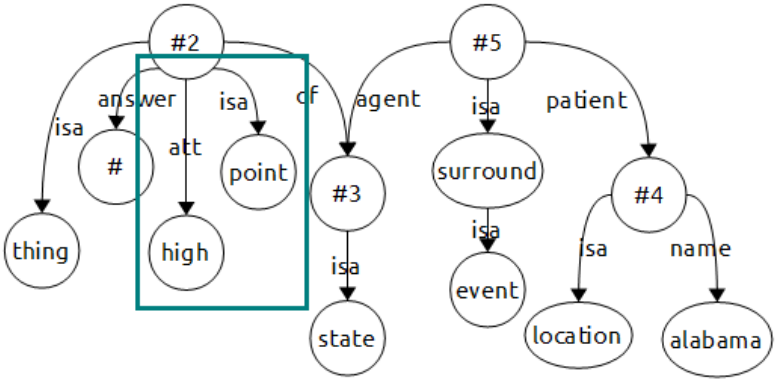


Figura 50: grafo del parsing della query 10

Come si nota il grafo risultate e' piu' complesso. A colpo d'occhio notiamo che possiamo applicare i pattern 1, 2 (sul nodo #2) e 4 (sul nodo #4), ma il pattern che vogliamo esaminare e' quello evidenziato. Si tratta di un pattern frequente quando nella frase viene inserito un soggetto ed un attributo legato ad esso. La risoluzione e' presentata sotto e consiste semplicemente nel fondere l'attributo con l'entita' a cui fa riferimento:

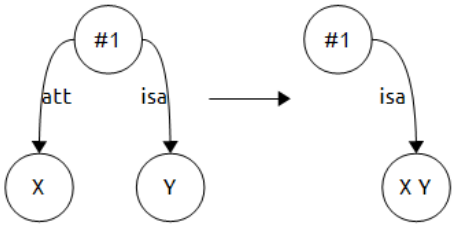


Figura 51: pattern 9

Ecco la risoluzione del grafo in esempio:

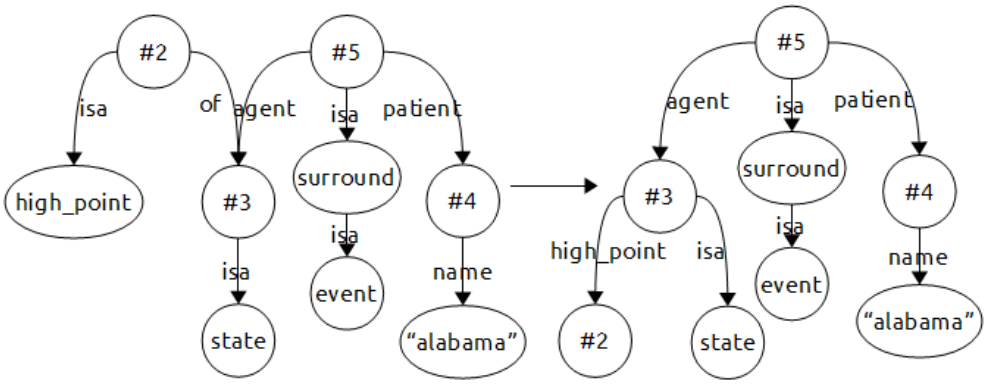


Figura 52: trasformazione della query 10

Al nodo #2 abbiamo applicato il pattern sopra descritto, trasformando cosi' il nodo "*point*" nel nodo "*high_point*". Successivamente abbiamo anche applicato il pattern 7, sempre al nodo #2. La trasformazione ottenuta e' ancora distante dal tipo di grafo con cui GeX puo' ricercare corrispondenze, infatti l'insieme di archi uscenti dal nodo #5 *agent, isa, patient* risulta essere un formalismo molto grammaticale. Per questo nel prossimo pattern ne vedremo la trasformazione.

Pattern n. 10

Prendiamo in considerazione un dettaglio della figura 49 proposta al paragrafo precedente.

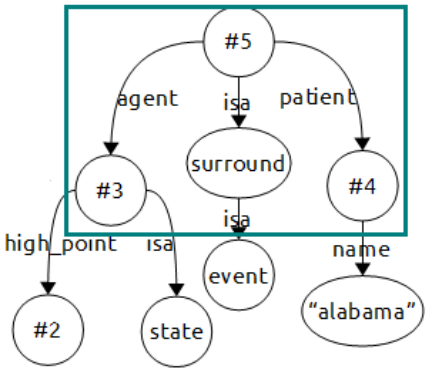


Figura 53: dettaglio della figura 49

Per semplificare questo e altri pattern dalla forma simile si ci deve concentrare sui nodi connessi tramite le proprieta' *agent* e *patient*: il primo nodo identifica l'*agente* dell'azione che si sta trattando nella frase, il secondo invece il partecipante alla frase su cui l'azione e' svolta. Facendo un esempio concreto nella frase "*Jack eat the lasagna.*" Jack identifica l'agente, la lasagna identifica il nodo patient.

Infine, quando ci troviamo di fronte ad una catena del tipo (*surround, isa, event*) ci troviamo in presenza del verbo o dell'azione che l'agente compie su patient. Considerando questi punti diciamo che se il nodo #3 compie l'azione *surround* sul nodo #4, quindi trasformiamo il nostro esempio come segue, dove concretamente il nodo #3 agisce sul nodo #4:

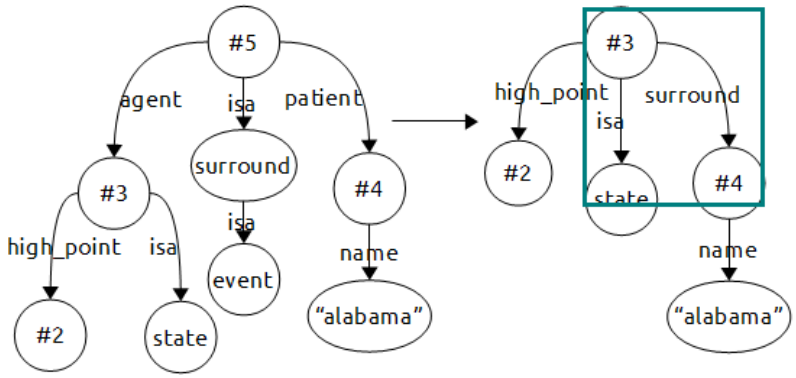


Figura 54: trasformazione della query 10

Il grafo puo' essere dato a GeX per l'interrogazione del dataset, previa modifica della label "*surround*" in "*borders*".

Di seguito la regola per il pattern 10.

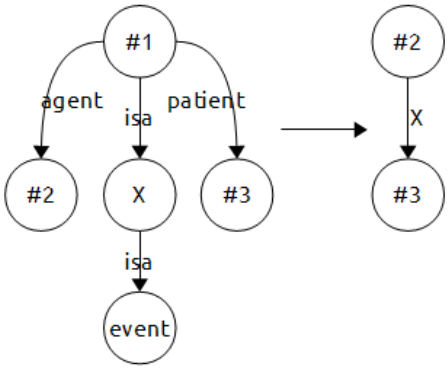


Figura 55: pattern 10

Pattern n. 11

Il prossimo pattern ha una struttura molto simile al pattern precedente:

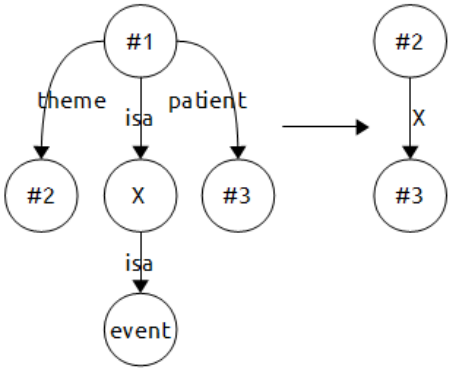


Figura 56: pattern 11

Come si vede l'unica differenza dal pattern 10 e' la presenza dell'arco *theme* anziche' *agent*. Questo e' dovuto al tipo di verbo che si utilizza nella nostra frase, ma il concetto e' sempre lo stesso. Presentiamo ora un esempio d'applicazione.

Query 11: "*Gli stati con una citta' chiamata "Akron".*" -> "*The states with a city named "Akron".*"

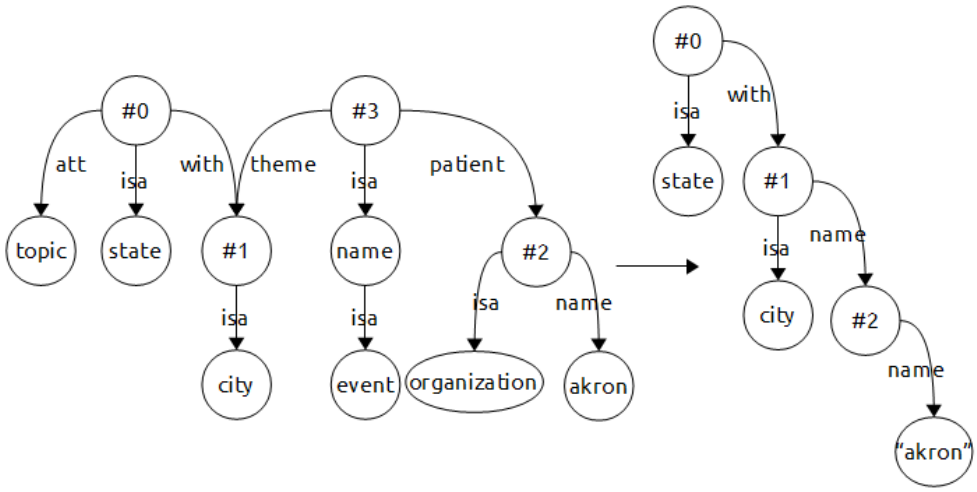


Figura 57: processo di semplificazione della query 11

In ordine sono stati applicati i seguenti pattern: pattern 3 sul nodo #0; pattern 4 sul nodo #2; pattern 11 sul nodo #3.

Il grafo che risulta necessita ancora di trasformazioni e verra' ripreso piu' avanti.

Presentiamo ora un esempio in cui sono presenti sia il pattern 10 che l'11. La query e' una leggera variazione della precedente.

Query 12: "*Gli stati che hanno una citta' chiamata "Akron".*" -> "*The states that have a city named "Akron".*"

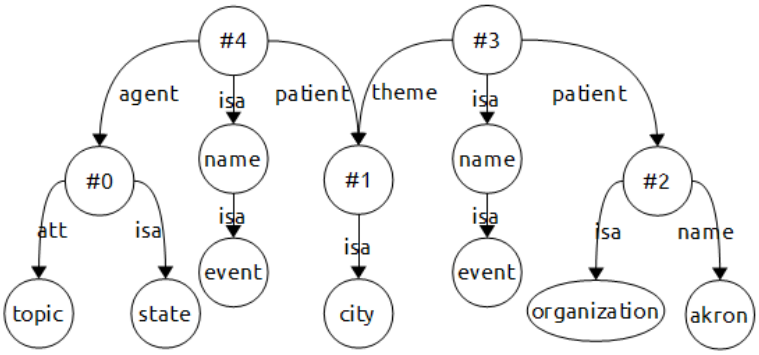


Figura 58: grafo del parsing della query 12

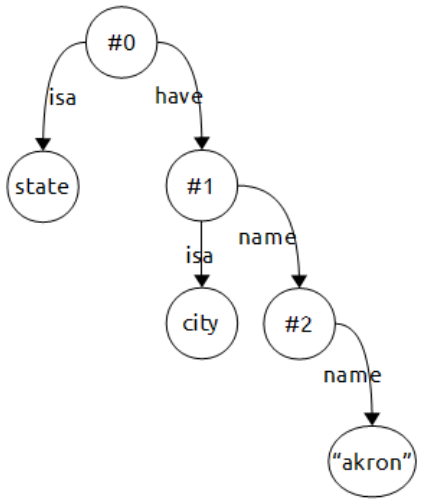


Figura 59: trasformazione della query 12

Sono stati applicati seguenti pattern: nodo #0 pattern 3; nodo #2 pattern 4; nodo #3 pattern 11 e nodo #4 pattern 10. Il grafo risultante necessita di altre trasformazioni.

Pattern n. 12

Il prossimo pattern si presenta nei casi in cui abbiamo un agente il cuo comportamento e' legato ad una appartenenza, ovvero quando compare la parola "*in*". Il pattern e' il seguente.

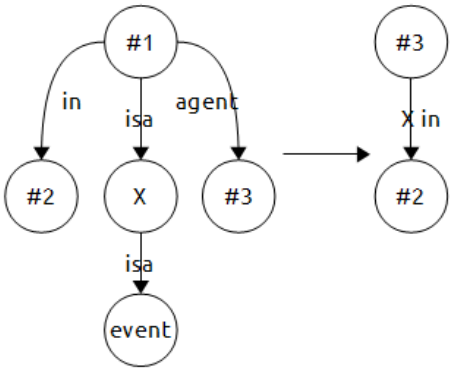


Figura 60: pattern 12

Ed ecco un esempio:

Query 13: "*Quali sono le citta' che si trovano in California?*" -> "*What are the cities which are in California?*"

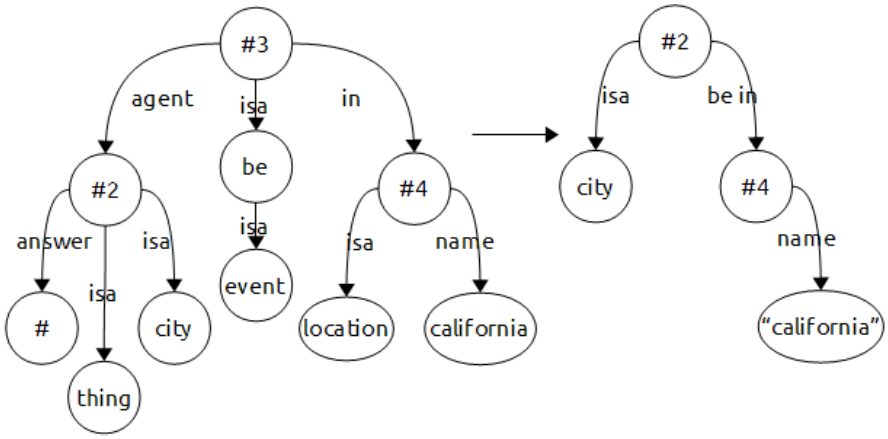


Figura 61: trasformazione query 13

Pattern applicati: nodo #2 pattern 1 e 2; nodo #3 pattern 12; nodo #4 pattern 4. Il grafo risultate e' pronto per essere utilizzato come query da GeX, previa una piccola modifica dell'etichetta "*be in*" in "*inState*".

Pattern n. 13

Il prossimo pattern e' legato alla presenza della parola "*number*" all'interno della frase e alla specifica struttura esposta sotto:

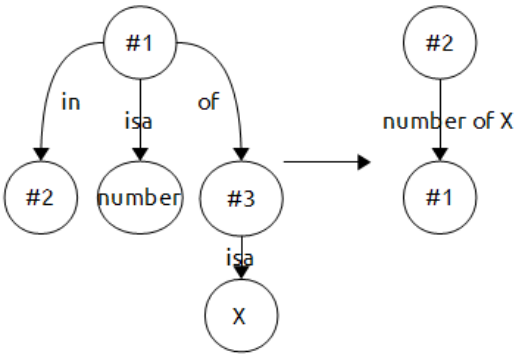


Figura 62: pattern 13

Query 14: "*Qual e' il numero si persone [che vivono] a Dallas*" -> "*What is the number of people in Dallas?*"

La query e' simile alla numero 8, infatti entrambe restituiscono a quanto ammonta la popolazione delle citta' in esame, pero' sono strutturate in modo diverso. Vediamo ora la risoluzione del nostro esempio.

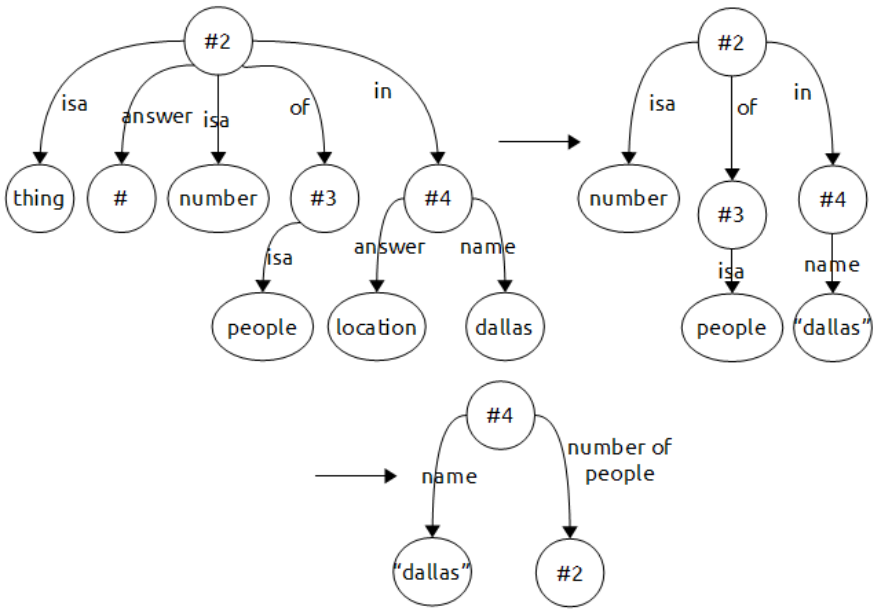


Figura 63: trasformazione query 14

Nel corso della trasformazione sono stati applicati i seguenti pattern: nodo #2 pattern 1 e 2; nodo #4 pattern 4; successivamente al nodo #2 e' stato applicato il pattern 13. Il grafo e' pronto per essere sottoposto a GeX, previa una trasformazione della label "*number of people*" in "*population*".

Pattern n. 14

Il prossimo pattern si presenta in diversi casi piu' specifici, osserviamo un esempio.

Query 15: "*Il punto piu' basso dell'Illinois.*" -> "*The lowest point in Illinois.*"

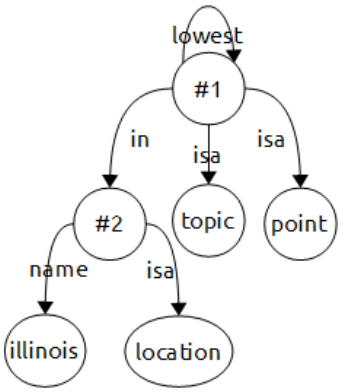


Figura 64: grafo parsing query 15

Notiamo subito il particolare arco che va da #1 in #1 etichettato con *lowest*; L'entita' #1 e' (*isa*) un punto (*point*) e ha la proprieta' di essere il piu' basso (*lowest*).

Proponiamo quindi la seguente semplificazione:

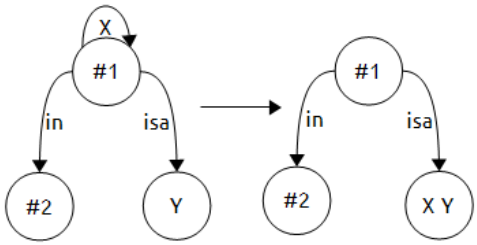


Figura 65: pattern 14

Applicando anche altri pattern la nostra query viene trasformata come segue:

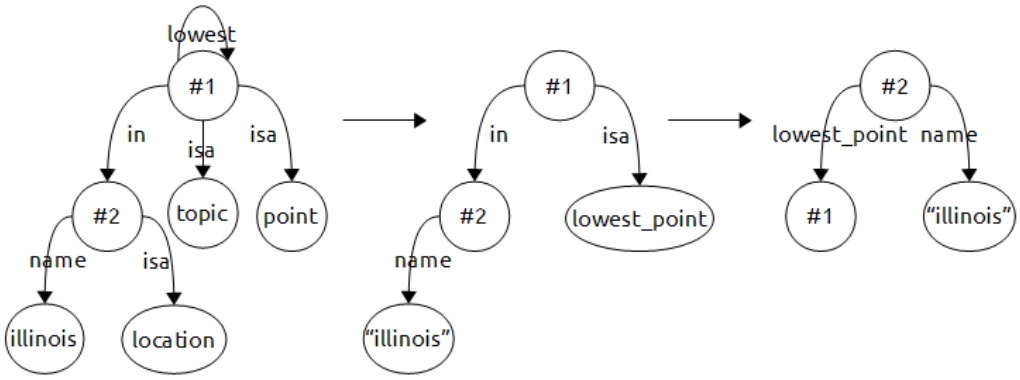


Figura 66: processo di trasformazione della query 15

Pattern applicati: nodo #1 pattern 3 e 14; nodo #2 pattern 4; successivamente abbiamo applicato al nodo #1 il pattern 8. Il grafo non necessita di altre modifiche e puo' essere utilizzato da GeX per interrogare il dataset.

Pattern n. 15

Il prossimo pattern si presenta in quelle interrogazioni in cui l'utente non e' in grado di definire che tipo di relazione intercorre tra due entita'. L'interrogazione e' del tipo "*l'entita' connessa in un qualche modo/relativa a un'altra entita*" ed ha la seguente struttura e semplificazion:

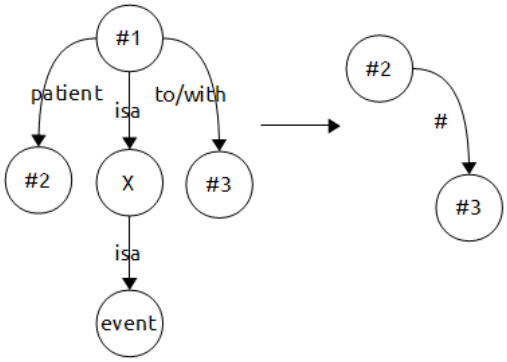


Figura 67: pattern 15

Specifichiamo che il nodo *X* puo' contenere qualsiasi label, per esempio "*connect*" o "*relate*" etc. La trasformazione di questo pattern porta ad avere una any label sull'arco che unisce i nodi interessati dalla relazione; tale label ci dice che sappiamo che i due nodi sono connessi ma non conosciamo la natura di tale connessione. Per estrarre i dati GeX sfruttera' anche gli altri attributi connessi ai nodi. Di seguito un esempio.

Query 16: "*Gli stati connessi ad un oggetto/entita' chiamata "Centerville".*" -> "*The states related to an object named "Centerville".*"

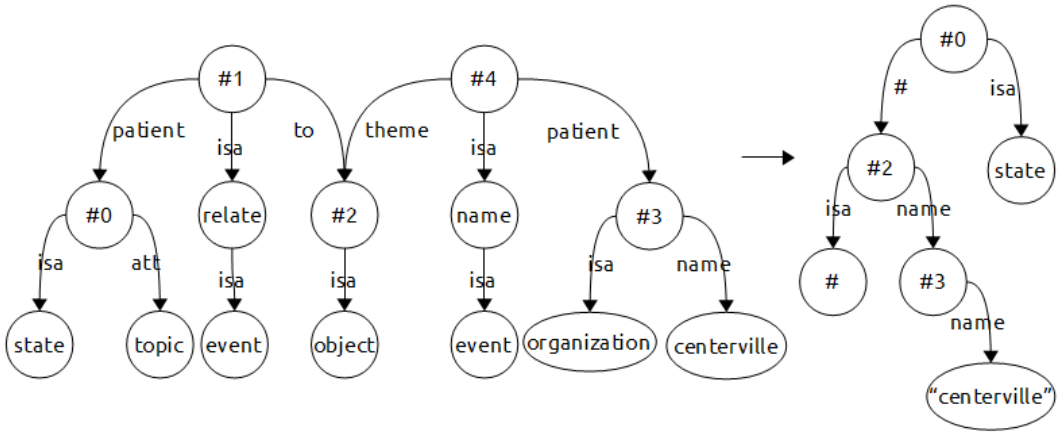


Figura 68: grafo del parsing della query 15

Il pattern 15 si trova a sinistra, nodo #1. Con ordine presentiamo i pattern che sono stati applicati: pattern 3 sul nodo #0; pattern 15 sul nodo #1; pattern 5 sul nodo #2; pattern 4 sul nodo #3; pattern 11 sul nodo #4. Il grafo risultate necessita di un'ulteriore trasformazione tramite il pattern 16 che andiamo a presentare.

Pattern n. 16

Osserviamo un dettaglio della trasformazione in figura 65:

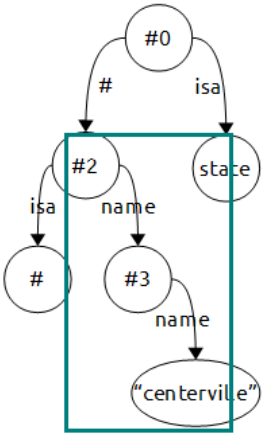


Figura 69: dettaglio trasformazione query 16

Facendo riferimento allo schema dei dati notiamo che un percorso (*#2, name, #3, name, "nome"*) non viene riconosciuto. Si introduce quindi questa breve semplificazione che collega il primo nodo del percorso direttamente con il nodo contenente il valore dell'attributo *name*.

Questo pattern e' anche rintracciabile in query precedente come la 11 e la 12.

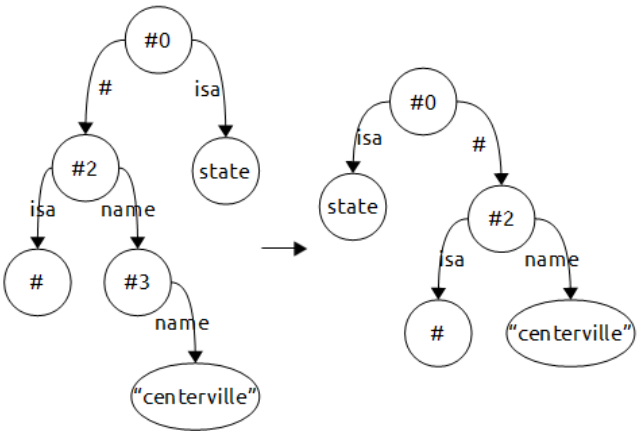


Figura 70: conclusione della trasformazione della query 16

Il grafo puo' ora essere utilizzato da GeX per interrogare lo schema dei dati. Di seguito lo schema del pattern:

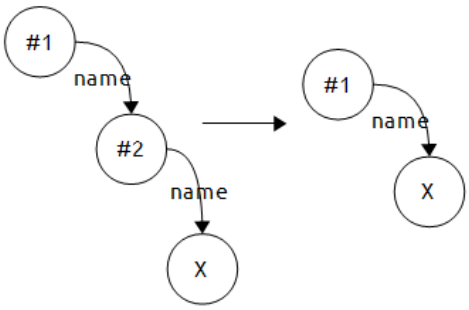


Figura 71: pattern 16

Pattern n. 17

Il diciassettesimo pattern si rintraccia nelle interrogazioni in cui si inserisce la parola "*through*" come "*I fiumi/le strade che attraversano lo stato*". E' un pattern un po' piu' specifico di quelli presentati fino ad ora, ma e' molto semplice da rintracciare e si presenta in diversi casi.

Query 17: "*Quali sono gli stati attraversati dal Potomac?*" -> "*What are the states that the Potomac runs through?*"

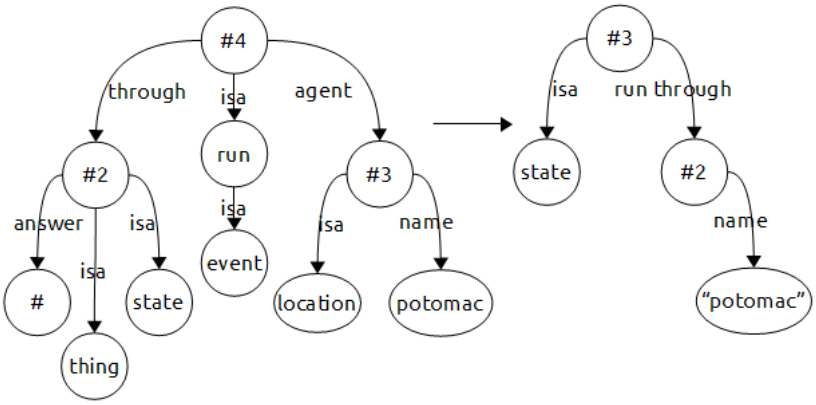


Figura 72: parsing della query 17 e trasformazione

Il pattern non dipende dalla parola contenuta nel nodo connesso tramite *isa* (qui *run*), infatti potrebbe essere *run*, *pass*, *go* etc.

Sono stati applicati i seguenti pattern: pattern 1 e 2 al nodo #2; pattern 4 al nodo #3 e pattern 17 al nodo #4. Il grafo ottenuto puo' essere

utilizzato come interrogazione da GeX previa trasformazione della label "*run through*" in "*flowsThrough*";

Ecco lo schema del pattern:

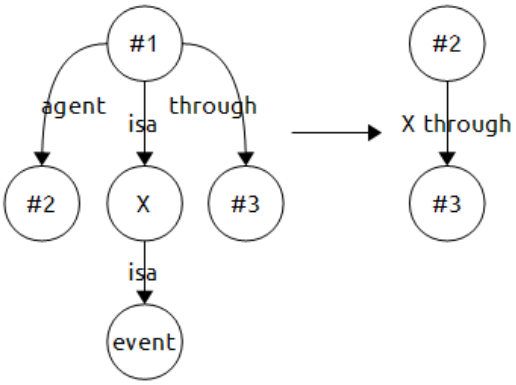


Figura 73: pattern 17

Pattern n. 18

Presentiamo ora un pattern particolare ma che e' stato riscontrato in diversi casi. Condizione necessaria per riconoscere tale pattern e' che le label dei nodi che contengono valori specifici di attributi siano racchiuse tra doppi apici ("): tale meccanismo permette di riconoscere in modo univoco il pattern in questione. Chiariamo meglio con alcuni esempi.

Query 18: "*Gli stati la cui abbreviazione e' "ar".*" -> "*The state abbreviated "ar".*"

La query in esame puo' sembrare un po' forzata ma e' necessario inserire un verbo per ogni frase, altrimenti il parser non lavora correttamente. Ulteriori dettagli piu' avanti nel paragrafo 3.4 sui pattern non risolti.

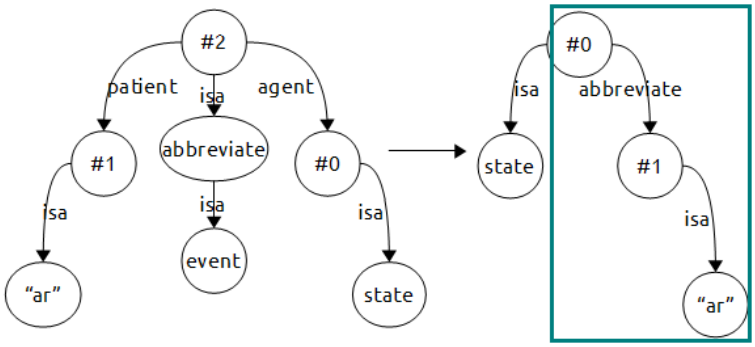


Figura 74: grafo parsing query 18 e semplificazione

Al grafo e' stato applicato solo il pattern 10 al nodo #2. A destra abbiamo evidenziato la struttura d'interesse. Un collegamento di quel tipo puo' essere letto come "*le entita' di tipo stato che hanno per abbreviazione un entita' di tipo "ar".*" e tale lettura, chiaramente, non genera alcun risultato perche' non esiste il tipo "ar"; il compito del pattern 18 e' proprio quello di reificare una struttura di questo tipo.

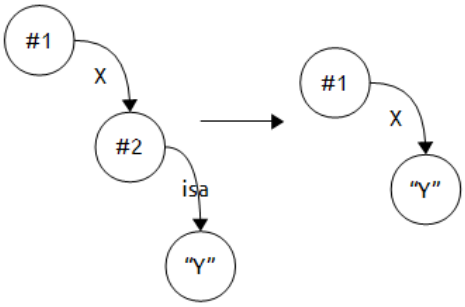


Figura 75: pattern 18

Quest'approssimazione necessita di alcune righe di spiegazione. Punti fondamentali del pattern sono il collegamento *isa*, che deve necessariamente esserci (altrimenti la struttura puo' essere gestita tramite altri pattern), e il fatto che il valore *Y* sia racchiuso tra doppi apici. Per quanto detto al paragrafo 3.2 sappiamo che quando ci troviamo davanti ad un nodo con label del tipo *"label"* significa che siamo in presenza del valore di un attributo specificato dall'utente, come un nome, un'abbreviazione, un

codice etc., ma nel contesto dello schema dei dati in uso il nodo contenente il valore della proprieta' non e' connesso al nodo padre (l'oggetto cui la proprieta' e' riferita) tramite un arco *isa*; si deduce quindi che il nodo #2 e l'arco *isa* non sono necessari e possono essere rimossi, collegando direttamente il nodo #1 tramite la proprieta' X al nodo contenente il valore di quella proprieta' per quel nodo. La semplificazione della query 18 puo' quindi essere conclusa.

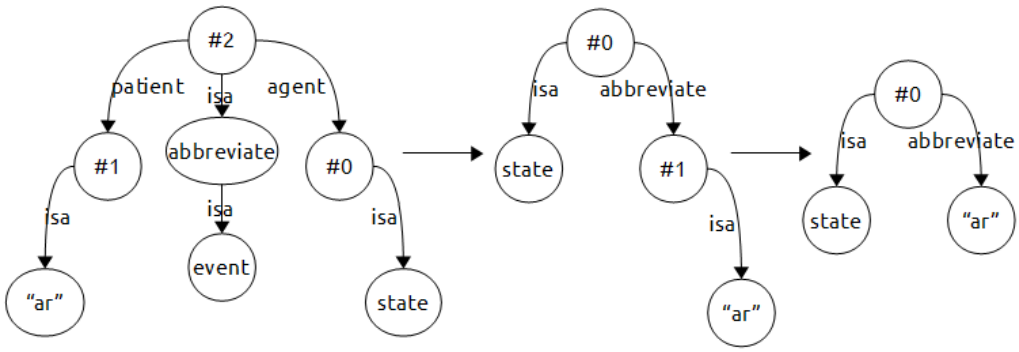


Figura 76: processo di semplificazione della query 18

3.4. Pattern non risolti

Riportiamo ora due pattern per i quali non e' stata trovata una semplificazione utile o univoca.

Pattern n. 19

Il primo pattern che esaminiamo si presenta nei casi in cui il parser non riesce a riconoscere tutti gli elementi in una frase e quindi non e' in grado di nominare correttamente alcuni archi; in questi casi il parser nomina gli archi con una stringa del tipo "*NN*" che sta ad indicare che due nodi sono si' in relazione, ma non si e' stabilita precisamente quale sia. Il pattern si verifica quando una frase non e' ben costruita o semplicemente quando Boxer non riesce a parsare correttamente.

Query 19: "*Qual e' la citta' capitale della Florida?*" -> "*What is the capital city in Florida?*"

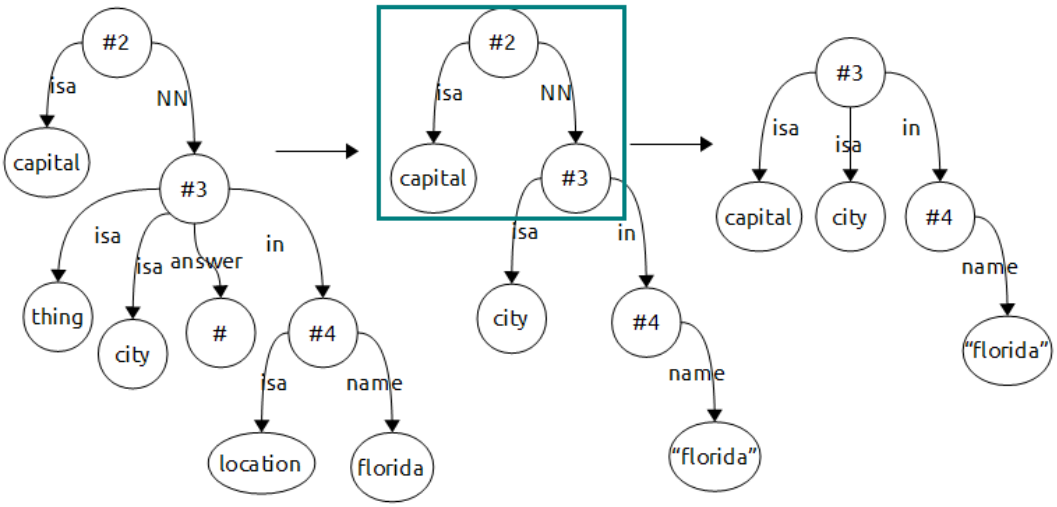


Figura 77: grafo del parsing della query 19 e parte di processo di trasformazione

Nell'immagine abbiamo applicato, al primo passaggio i pattern 1 e 2 sul nodo #3 ed il pattern 4 sul nodo #4. Al secondo passaggio evidenziamo il pattern in questione. In questo caso specifico contiene delle informazioni

relative alla tipologia di citta' che si sta esaminando. Al terzo passaggio abbiamo proposto un trasformazione che segue il seguente pattern:

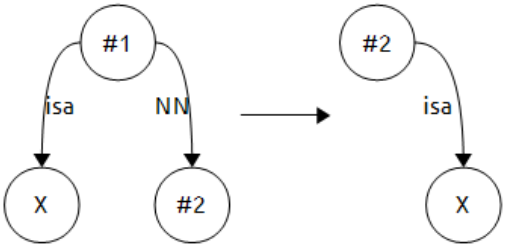


Figura 78: possibile pattern 19

L'obiettivo del pattern sarebbe quello di eliminare l'arco *NN* aggiungendo l'attributo *X* al nodo #2. Nella query 19 potremmo applicare il pattern 8 ma ci troviamo davanti ad un conflitto: su quale ramo (*isa, attributo*) applichiamo il pattern? Presentiamo entrambe le casistiche:

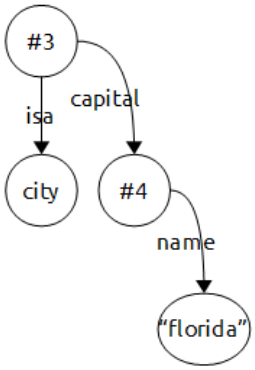


Figura 79: possibile trasformazione query 19

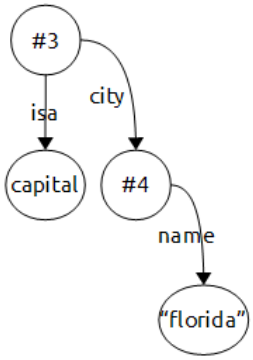


Figura 80: possibile trasformazione query 19

In entrambi i casi i grafi, sottoposti a GeX, non trovano corrispondenze. La figura 76 puo' essere letta come "*Un'entita' di tipo citta' che e' capitale/ha capitale di un'entita' chiamata Florida.*": questa query e' formalmente corretta ma non restituisce risultati perche' la proprieta' *capital* e' legata agli oggetti di tipo *state*, non a quelli di tipo *citta'*: sono gli stati ad avere capitali, non le citta' che hanno la proprieta' di essere

capitale di uno stato. Per quanto concerne la figura 77 invece il grafo non genera alcuna risposta dato che non esiste la classe tipo *capital*.

In altri casi invece il pattern 19 porta a perdita di informazioni come nell'esempio che segue.

Query 20: "*I luoghi che confinano con il Nebraska e che hanno una citta' chiamata "Casper".*" -> "*The locations border Nebraska and that have a city named "Casper".*"

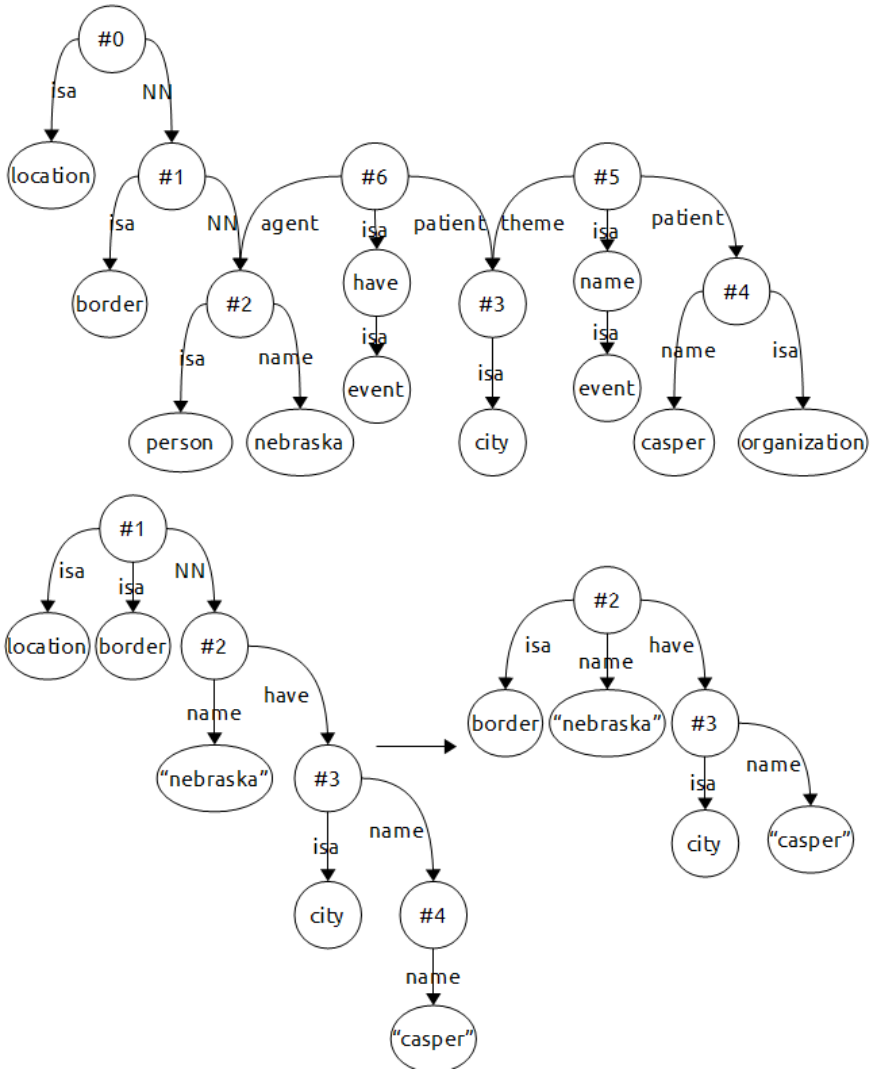


Figura 81: processo di trasformazione della query 20

Al primo passaggio abbiamo applicato il pattern 18 al nodo #0, il pattern 4 al nodi #2 e #4, il pattern 10 al nodo #6 e il pattern 11 al nodo #5; successivamente abbiamo applicato il pattern 19 al nodo #1 (con perdita di informazioni) e il pattern 16 al nodo #3

Come si vede, tra il secondo e il terzo passaggio abbiamo perso l'informazione relativa a *location*; inoltre all'ultimo passaggio non si puo' procedere oltre con le trasformazioni e il grafo non genera risultati: infatti non esiste la classe *border*, ma soprattutto leggendo il grafo notiamo che stiamo cercando "*un'entita' chiamata Nebraska che ha una citta' chiamata Casper.*" E questo non corrisponde a quello che volevamo esprimere con la query.

Come abbiamo accennato la presenza di questo pattern puo' essere dovuta a errori di parsing da parte di Boxer: il che non significa che la frase in linguaggio naturale sia scritta male, ma piuttosto che il software non e' in grado di stabilire la natura della connessione tra due archi perche' non parsa correttamente alcuni tipi di frasi. Questo e' il caso di alcune query del tipo "*The state with the abbreviation "ar".*": il grafo risultante di questa query contiene il pattern 19; se pero' diamo al parser la frase "*The states abbreviated "ar".*" (query 18, figura 71) il grafo risultante e' corretto e non presenta il pattern 19 e puo' essere facilmente semplificato. Il consiglio e' quindi quello di rivedere la frase in linguaggio naturale e manipolarla per fornire a Boxer piu' dettagli, solitamente la causa e' la mancanza di un verbo.

Pattern n. 20

Il pattern che esaminiamo ora e' collegato alle interrogazioni in cui compare il termine "*with*" o "*to*", ad esempio "*Lo stato con una citta' chiamata Providence.*" etc. Per esaminarlo consideriamo una query gia' presa in esame.

Query 11: "*Gli stati con una citta' chiamata Akron*" -> "*The states with a city named Akron.*"

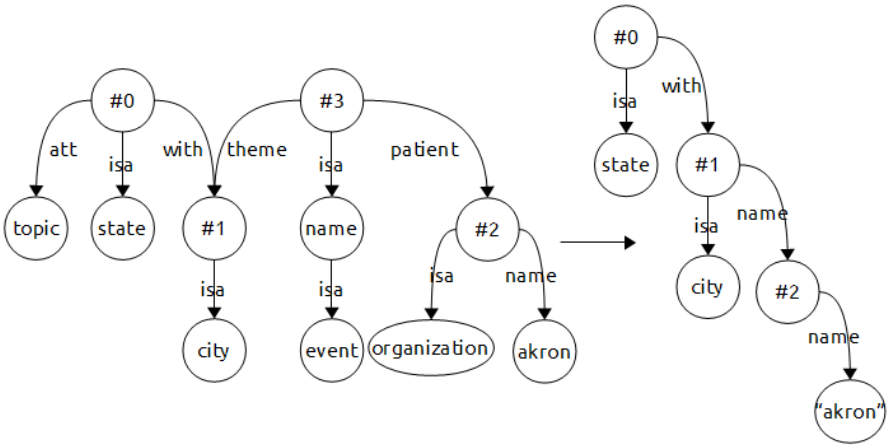


Figura 82: processo di semplificazione della query 11

Al primo passaggio sono stati applicati il pattern 3 sul nodo #0, il 4 sul nodo #2 e il pattern 11 sul nodo #3, ottenendo il risultato a destra. Tale risultato puo' essere ulteriormente semplificato tramite l'applicazione del pattern 16 sul nodo #1. A questo punto osserviamo il percorso (*#0, with, #1*). Lo schema che vorremmo ottenere ha quel particolare arco etichettato con la parola *hasCity*, ma l'applicazione di tale label dipende fortemente dal contenuto dei nodi presenti nel grafo. Una possibile risoluzione di questo pattern puo' essere quella di sostituire l'etichetta *with* con una any label *#* in modo da generalizzare nella ricerca del percorso. Una soluzione del genere resta essere un puo' "bruta" e non sempre le query possono dare risposte.

3.5. Analisi di query interrogative

Nei capitoli precedenti abbiamo spesso mostrato query in forma interrogativa che iniziano con "*What*". Boxer e' in grado di parsare correttamente questo tipo di frase, ma non si puo' dire altrettanto della query che iniziano con "*How*" o con "*Where*"; i grafi risultati dal parsing di frasi di questo tipo sono spezzati, ovvero uno o piu' nodi sono separati dal resto della struttura. Di seguito alcuni esempi.

Query 21: "*Quant'e' alto il monte Mckinley?*" -> "*How high is mount Mckinley?*"

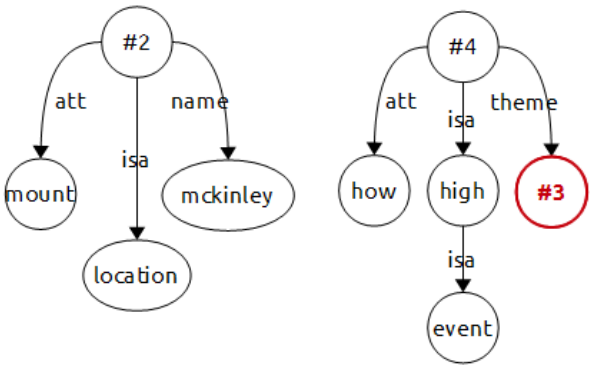


Figura 83: parsing della query 21

Query 22: "*Dove si trova Indianapolis?*" -> "*Where is Indianapolis?*"

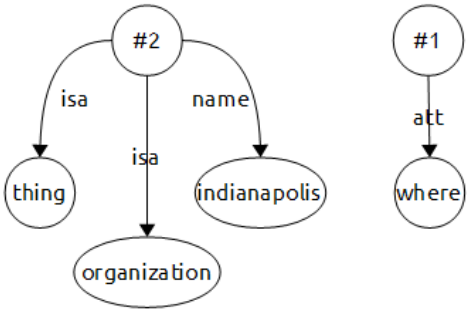


Figura 84: parsing della query 22

Come si nota i grafi sono in entrambi i casi completamente divisi; inoltre, nella figura 80, abbiamo evidenziato in rosso il nodo #3 che non e' dichiarato nella parte iniziale del file risultante dal parsing (ovvero dove tutti i nodi sono dichiarati) ma appare nella zona in cui sono definiti i collegamenti tra i nodi senza nessuna dichiarazione.

Caso diverso e' quello riguardante le domande che iniziano con "*Which*": in alcuni casi il parser lavora correttamente e genera un grafo esatto e trasformabile (si veda la figura 82); in altri casi i grafi che vengono generati sono ben strutturati ma errati semanticamente (si veda la figura 83); infine puo' capitare che Boxer generi dei grafi divisi come negli esempi precedenti (figura 84).

Query 23: "*Quali laghi si trovano in Minnesota?*" -> "*Which lakes are in Minnesota?*"

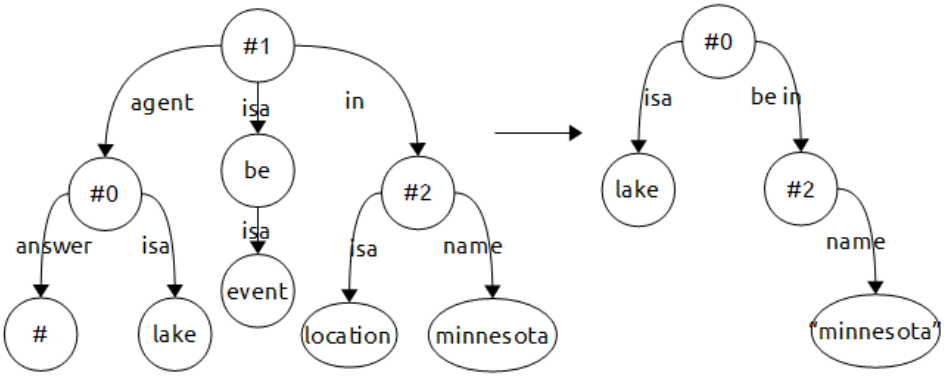


Figura 85: grafo del parsing della query 23 e relativa trasformazione

Per semplificare il grafo sono stati applicati i pattern 2 al nodo #0, il pattern 12 al nodo #1 e il pattern 4 al nodo #2; il grafo ottenuto e' utilizzabile da GeX previa modifica della label dell'arco "*be in*" in "*inState*".

Query 24: "*Quali capitali ci sono negli stati che confinano con il Nebraska?*" -> "*Which capitals are in the states that border Nebraska?*"

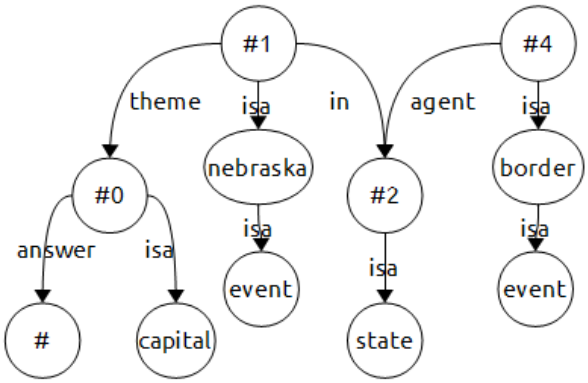


Figura 86: grafo del parsing della query 24

Il grafo non ha senso, lo si vede nello specifico del percorso (*Nebraska, isa, event*): *nebraska* non e' un verbo o un'azione, e' il nome di uno stato.

Query 25: "*In quale stato si trova Kalamazoo?*" -> "*Which states is Kalamazoo in?*"

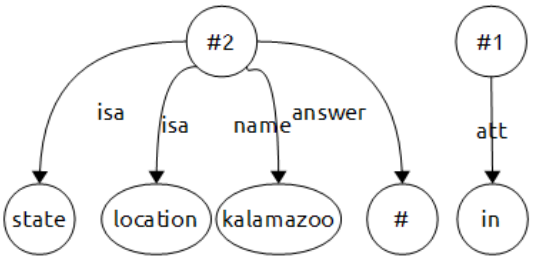
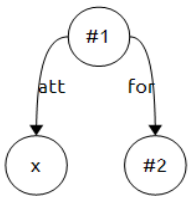
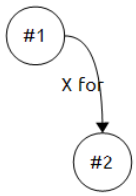
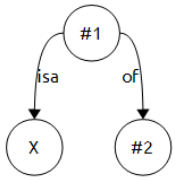
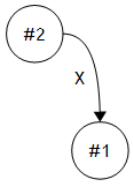
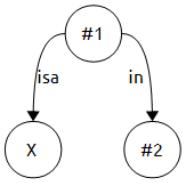
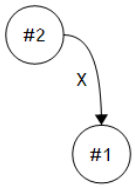
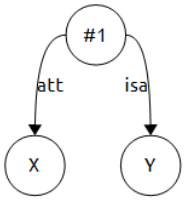
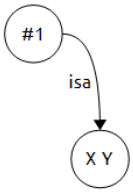
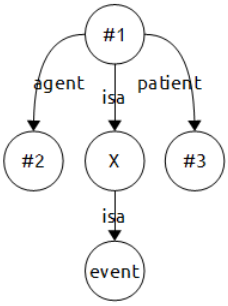
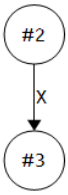
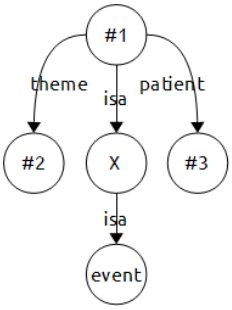
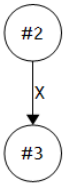


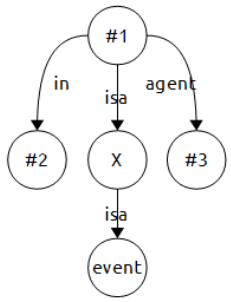
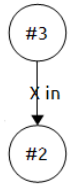
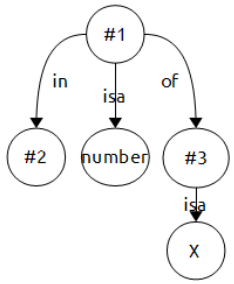
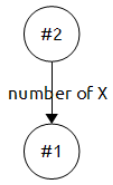
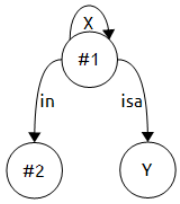
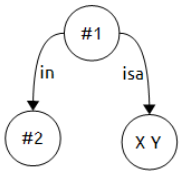
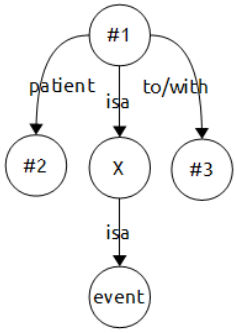
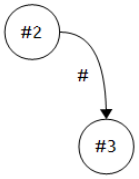
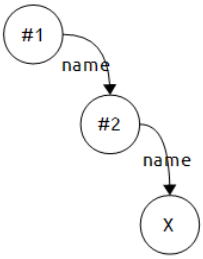
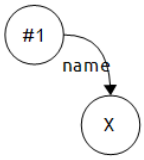
Figura 87: grafo del parsing della query 25

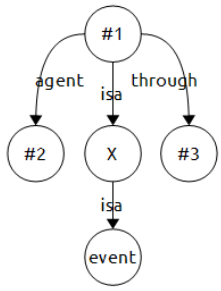
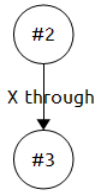
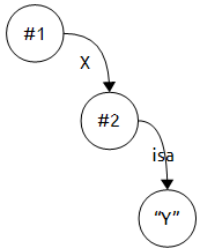
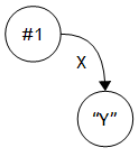
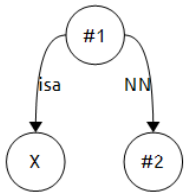
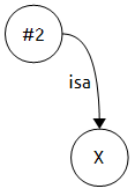
3.6. Tabella riassuntiva dei pattern rintracciati

Di seguito proponiamo una tabella riassuntiva contenente i pattern rintracciati e la corrispondente semplificazione. Per maggiori dettagli si rimanda alla sezione del pattern specifico.

Riferimento	Struttura	Risoluzione
Pattern n. 1	#1 —isa→ thing	✕
Pattern n. 2	#1 —answer→ #	✕
Pattern n. 3	#1 —att→ topic	✕
Pattern n. 4	#1 —isa→ #2; #1 —name→ X Dove #2 puo' essere: "location", "organization", "person" o "entity".	#1 —name→ "X"^^xsd:string
Pattern n. 5	#1 —isa→ entity	✕ #1 —isa→ #

Pattern n. 6		
Pattern n. 7		
Pattern n. 8		
Pattern n. 9		
Pattern n. 10		
Pattern n. 11		

Pattern n. 12		
Pattern n. 13		
Pattern n. 14		
Pattern n. 15		
Pattern n. 16		

Pattern n. 17		
Pattern n. 18		
Pattern n. 19		

Capitolo 4

Prove sperimentali

In questo capitolo verranno presentate alcune prove sperimentali interessanti utili per completare l'analisi compiuta nell'ambito di questa tesi. L'obiettivo e' quello di verificare la correttezza dei pattern presentati al capitolo precedente, ma anche quello di capire quali sono le situazioni e le motivazioni che possono portare al mancato successo nell'interrogazione dei dati, situazione preceduta dal processo di parsing con Boxer e trasformazione delle query tramite i pattern.

Il capitolo racchiude le prove sperimentali eseguite su due collezioni differenti: geobase, schema utilizzato fino ad ora, e DBLP, collezione presentata al capitolo 2, paragrafo 2.1 che contiene informazioni riguardanti le pubblicazioni letterarie di vario tipo. Al termine presenteremo un'analisi dei dati raccolti.

4.1. Premesse

Verranno presentati due set di prove: il primo eseguito sulla stessa collezione di dati su cui sono stati ricercati i pattern, ovvero il file RDF

geobase, il secondo set sulla collezione presentata al paragrafo 2.1 contenete informazioni riguardo a pubblicazioni di vario tipo.

Per ogni prova il procedimento seguito e' lo stesso:

- Passo 1: la query in lingua inglese e in linguaggio naturale viene sottoposta a Boxer;
- Passo 2: il grafo del parsing viene semplificato (se possibile) tramite l'utilizzo dei pattern esposti al capitolo precedente;
- Passo 3: il grafo ottenuto viene sottoposta a GeX e si valutano le risposte del software;
- Passo 4: conclusioni tratte dallo studio della prova.

Per ogni prova verranno riportasti i pattern applicati e su che nodo.

Le regole sulla sintassi da utilizzare con il software Boxer sono riportati al capitolo precedente, paragrafo 3.2, e saranno rispettate anche qui.

I pattern verranno sempre applicati in ordine crescente, dal nodo con identificatore *#numero* piu' piccolo al piu' grande.

Di seguito una tabella contenente le query utilizzate nei prossimi paragrafi e il riferimento alla loro posizione; dalla query n. 1 alla n. 9 ci riferiamo al dataset geobase; dalla n.9 al termine sono le query eseguite su DBLP.

n. query	Query	Riferimento alla query
1	"The locations bordering Massachusetts that have a city named "Warwick"."	Query n. 1
2	"What are the roads in Arizona?"	Query n. 2
3	"The states that have the mountain named "Sanford"."	Query n. 3

4	"The roads that go through Arizona."	Query n. 4
5	"Which mountains are in Washington?"	Query n. 5
6	"The code of Nevada."	Query n. 6
7	"The area of lake Winnembago."	Query n. 7
8	"The area of lake named "Winnembago"."	Query n. 8
9	"The river length "1638"."	Query n. 9
10	"What are the phdthesis?"	Query n.10
11	"The articles of year "1993"."	Query n. 11
12	"The books written in "1988"."	Query n. 12
13	"The pages of the entity titled "Programmed Control of Asynchronous Program Interrupts."."	Query n. 13
14	"The object published by Benjamin/Cummings."	Query n. 14
15	"The books which are editing by Yasushi Kiyoki."	Query n. 15
16	"What are the object connected to a person named "Subrata Dasgupta"?"	Query n. 16
17	"The titles of the object created by a person who has also created an article named "Statistical Processors."."	Query n. 17
18	"The objects created by someone named "Roberto Giacobazzi"."	Query n. 18

4.2. Prove sperimentali su geobase

Query n. 1

"I luoghi che confinano con il Massachusetts e che hanno una citta' chiamata "Warwick"." -> "The locations bordering Massachusetts that have a city named "Warwick"."

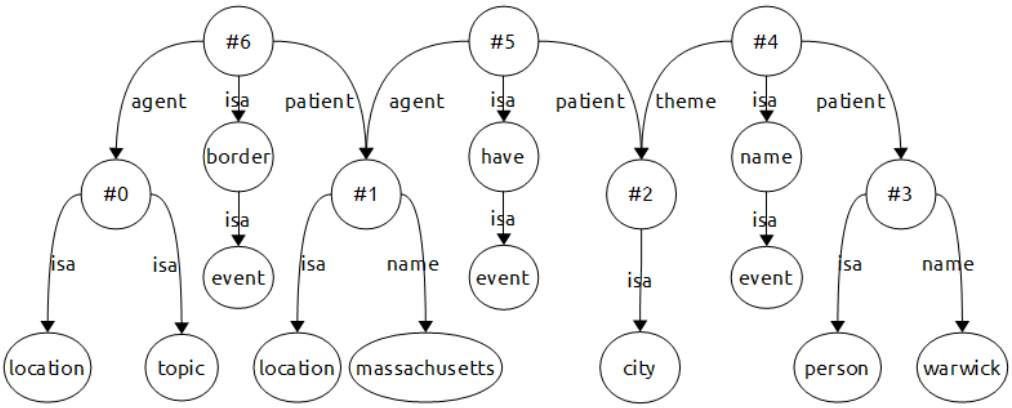


Figura 88: grafo del parsing della query 1

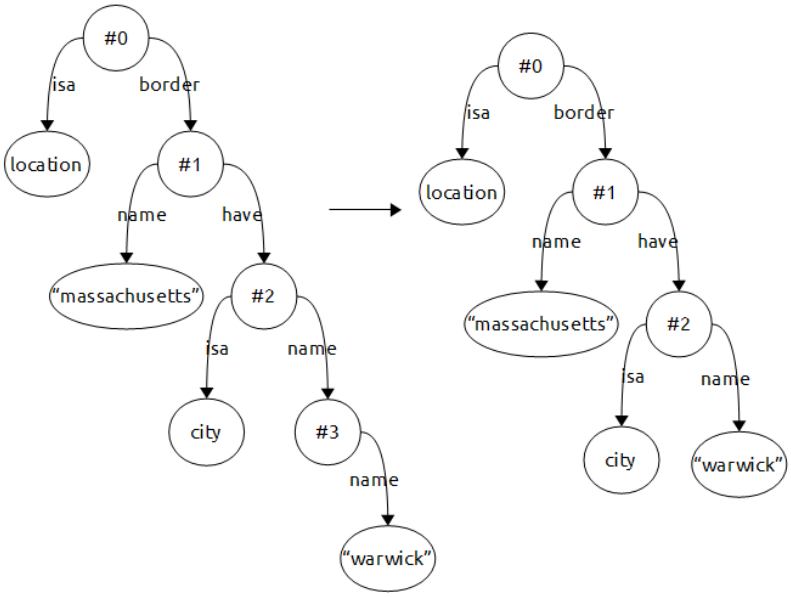


Figura 89: processo di semplificazione della query 1

Pattern applicati per la trasformazione: pattern 3 sul nodo #0; pattern 4 sul nodo #1 e #3; pattern 11 sul nodo #4; pattern 10 sul nodo #5 e #6. Al secondo passaggio e' stato applicato il pattern 16 al nodo #2.

Il grafo puo' essere usato da GeX, previa modifica delle label "*border*" in "*borders*" e "*have*" in "*hasCity*"; GeX pero' non fornisce alcuna risposta all'interrogazione del database tramite questa query. Il processo di parsing e la query risultante sono sintatticamente corretti, ma se leggiamo il grafo ottenuto abbiamo una frase del tipo "*Un'entita' generica che confina con il*

Massachusetts e che (il Massachusetts) ha una citta' chiamata "Warwick".": il punto focale dell'errore e' il fatto che la citta' di Warwick e' intesa come appartenente allo stato del Massachusetts, non come informazione legata alla location che vogliamo trovare, cosa che noi invece intendevamo. Il grafo quindi non generera' mai alcuna risposta dato che Warwick e' nello stato di Rhode Island.

Questa query mette in evidenza un limite del parser: Boxer fatica a parsare in modo corretto frasi piu' estese e complesse dove si fa uso della coordinazione; cio' che non riesce a trattare correttamente e' il concetto di "*questo oggetto ha questa proprieta' e allo stesso tempo anche questa.*". Se leggiamo il grafo in figura 85 notiamo che il problema e' gia' presente subito dopo il parsing: al centro abbiamo l'entita' #1 che e' connessa all'entita' #2 tramite il paradigma agent-patient, ovvero #1 agisce su #2; l'agire e' costituito dalla proprieta' di appartenenza (*have*), quindi #1 *ha* un'entita' #2; l'entita' #1 rappresenta la location Massachusetts, e l'entita' #2 un elemento di tipo city, ovvero la citta' chiamata Warwick. Ne deriva che subito dopo il parsing la struttura della frase non e' piu' quella espressa inizialmente.

Abbiamo provato a formulare la query in molte varianti come "*The locations border Massachusetts having a city named Warwick*", ma il grafo generato e' simile a quello in figura 85. Un'altra forma prevedeva la sostituzione di "*border*" con "*adjacent to*" ma viene generato un grafo non completamente connesso, quindi non utilizzabile; infine un altro esempio e' la query 20 del capitolo 3.4: "*The locations border Nebraska and that have a city named "Casper".*"; la struttura risultante non dipende dai nomi delle localita' inserite ed e' presentata in figura 78, di cui riportiamo una parte qui sotto per semplicita'. Come si vede il grafo presenta il pattern 19 per il quale non e' stata definita una risoluzione specifica.

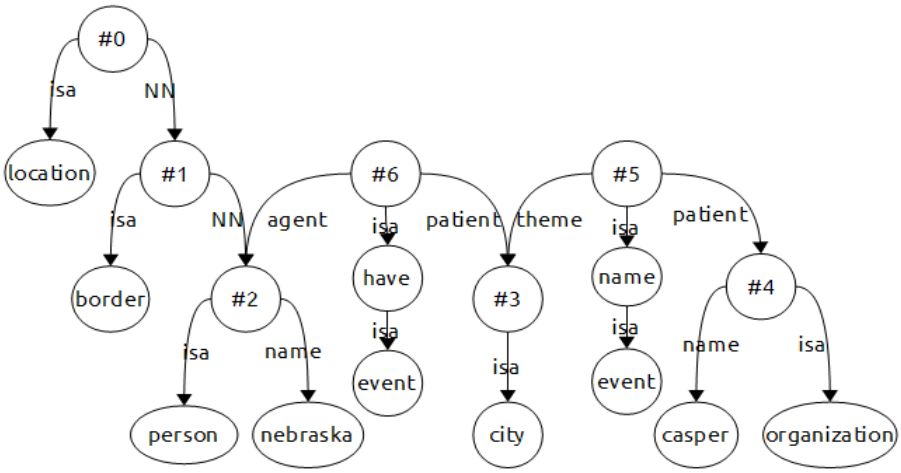


Figura 90: grafo del parsing della query 20 al paragrafo 3.5

La conclusione che si trae dall'analisi di questo tipo di interrogazione e' che Boxer non e' sempre in grado di rappresentare la coordinazione che esiste tra gli elementi di una frase in linguaggio naturale nel grafo del parsing.

Query n. 2

"*Quali sono le strade in Arizona?*" -> "*What are the roads in Arizona?*"

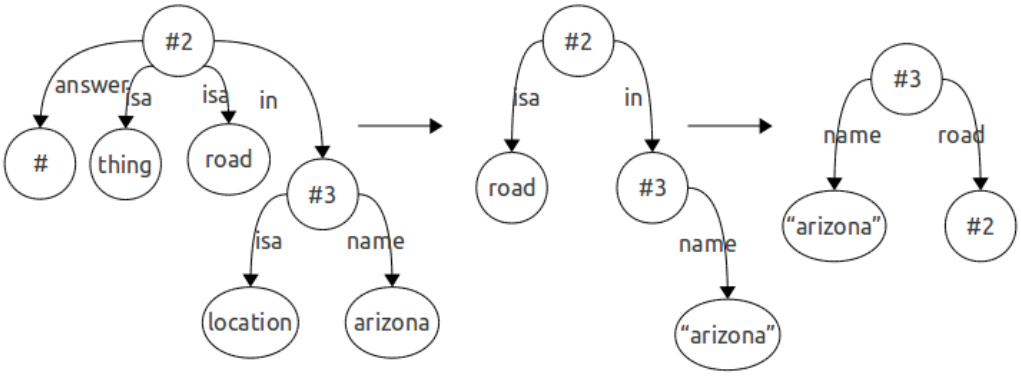


Figura 91: parsing della query 2 e processo di semplificazione

Al grafo sono stati applicati i seguenti pattern: pattern 1 e 2 al nodo #2; pattern 4 al nodo #3; successivamente al nodo #2 e' stato applicato il pattern 8.

Il grafo risultante sottoposto a GeX non fornisce alcuna corrispondenza: non esistono infatti archi con label *road*. Anche sostituendo la label con *passessThrough* non otterremmo risultati in quanto sono le strade ad avere la proprieta' di "passare attraverso" uno stato, non gli stati che hanno una proprieta' di "essere attraversati".

Nel caso di questa query e di altre del tipo "*The mountain in Colorado*" o "*What are the lakes in the state of Michigan?*" il problema risiede nel fatto che durante il parsing il soggetto principale della query si sposta dall'essere la strada, il lago o la montagna all'essere l'entita' che identifica lo stato; ma gli stati, come si vede nello schema dei dati presentato al capitolo 2 paragrafo 2.1, non hanno la proprieta' di avere montagne, avere laghi o essere attraversati da strade: essi sono connessi agli oggetti naturali citati tramite la proprieta' *inState* che gli stessi oggetti hanno e che, idealmente, e' visibile dall'oggetto naturale ma non dallo stato.

Query n. 3

"*Gli stati che hanno una montagna chiamanta "Sanford".*" -> "*The states that have the mountain named "Sanford".*"

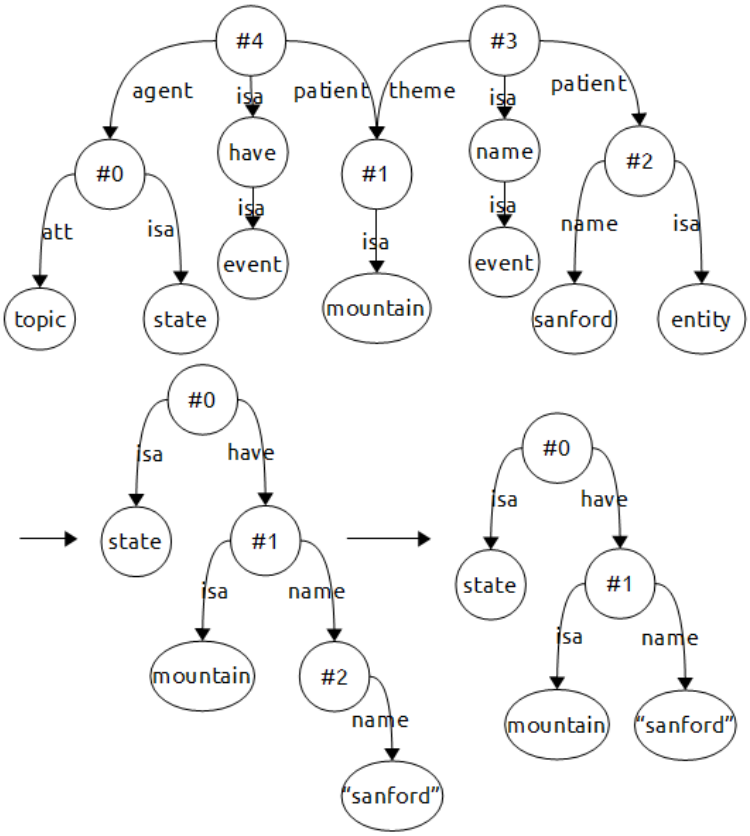


Figura 92: grafo del parsing della query 3 e processo di semplificazione

Al primo passaggio sono stati applicati i seguenti pattern: nodo #0 pattern 3; nodo #2 pattern 4; nodo #4 pattern 10 e nodo #5 pattern 11; al secondo passaggio abbiamo applicato il pattern 16 al nodo #2.

Il grafo risultante e' sintatticamente corretto e rispetta a pieno il senso della query fornita a Boxer, ma sottoposta a GeX non vengono trovate corrispondenze nell'insieme dei dati. In questa query infatti ritroviamo il problema esposto alla query precedente e oservando il grafo notiamo come si stia cercando uno stato che ha la proprieta' di "avere" una montagna; cosa impossibile dato che gli stati, come si vede dallo schema dei dati, non hanno tale proprieta'.

Query n. 4

"*Le strade che attraversano l'Arizona.*" -> "*The roads that go through Arizona.*"

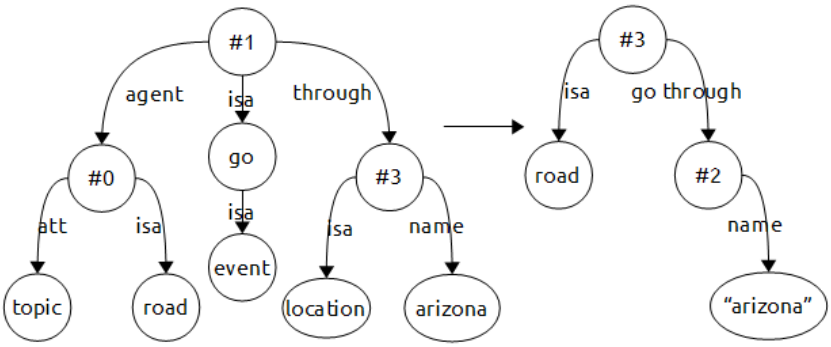


Figura 93: grafo del parsing e trasformazione della query 4

Al grafo sono stati applicati il pattern 3 al nodo #0, il pattern 4 al nodo #3 ed il pattern 17 al nodo #1. Il grafo risultante puo' essere sottoposto a GeX previa modifica della label "*go through*" in "*passessThrough*".

Interrogando il sistema con questo grafo GeX estrae con successo le strade che attraversano l'Arizona, ovvero la road40, la road 8, la road15 e la road10.

Questa query e' molto simile alla query 2, ma le leggere differenze di costruzione della frase in linguaggio naturale mettono Boxer nella posizione di creare un grafo linguistico che ha come soggetto la strada e non lo stato che attraversa. In questo modo dalla strada tramite la proprieta' *passessThrough* possiamo estrarre in modo corretto i dati riguardanti gli stati.

Query n. 5

"*Quali montagne si trovano a Washington?*" -> "*Which mountains are in Washington?*"

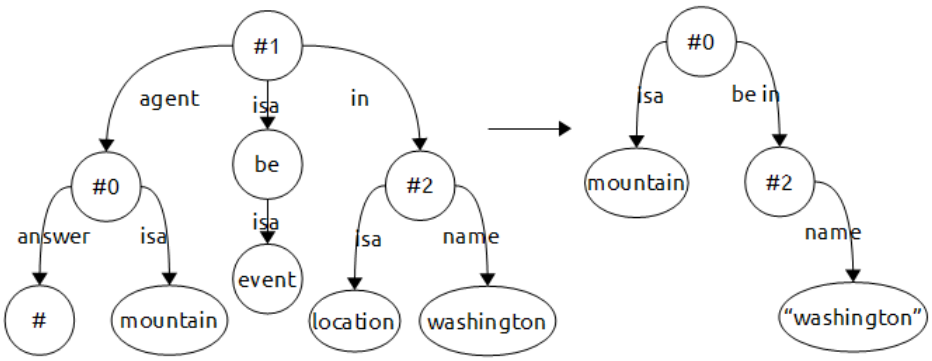


Figura 94: grafo del parsing e trasformazione della query 5

Il grafo e' stato sottoposto ai seguenti pattern: pattern 2 sul nodo #0; pattern 4 sul nodo #2; pattern 10 sul nodo #1. Prima di sottoporre il grafo a GeX modifichiamo la label "*be in*" in "*inState*"; il software estrae con successo le istanze richieste, ovvero il monte Rainier.

La query e' simile alla n. 3 ma anche in questo caso, come nel precedente, la variazione della struttura dell'interrogazione in linguaggio naturale ci permette di estrarre con successo i dati; il focus della nostra query e' infatti concentrato sull'entita' di tipo *mountain* e da qui andiamo a ricercare le entita' di tipo stato.

Gia' dopo poche prove risulta evidente come piccole variazioni nella costruzione di un'interrogazione in linguaggio naturale possano portare ad estrarre con successo o insuccesso i dati. Risulta anche evidente come per molte delle query che portano ad insuccessi nell'interrogazione del database esistano delle query dal significato identico ma costruite in modi linguistici diversi che portano ad estrarre correttamente i dati.

Query n. 6

"*Il codice del Nevada.*" -> "*The code of Nevada.*"

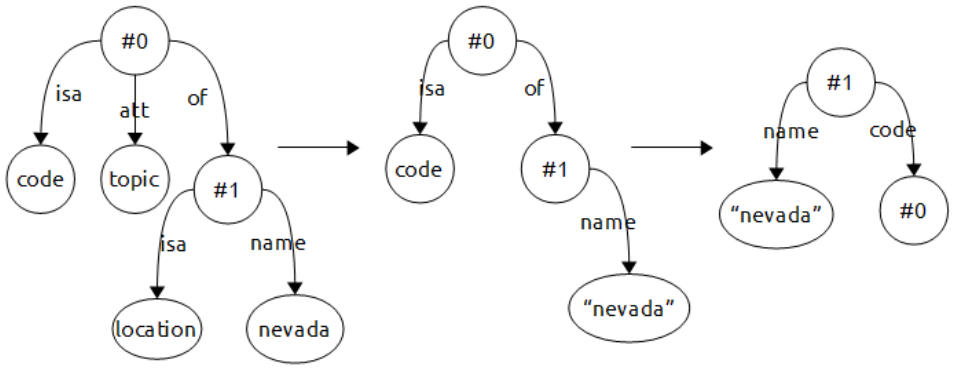


Figura 95: grafo del parsing e trasformazione della query 6

Al grafo risultante dal parsing di Boxer sono stati applicati il pattern 3 al nodo #0 ed il pattern 4 al nodo #1; successivamente e' stato applicato il pattern 7 al nodo #0. Il grafo risultante puo' essere sottoposto a GeX senza ulteriori modifiche ed il software estrae con successo l'istanza richiesta, ovvero il codice 36 dello stato del Nevada.

Questa query ci permette di evidenziare il buon funzionamento dei pattern indipendentemente dal contenuto dei nodi istanza.

Query n. 7

"*L'area del lago Winnembago*" -> "*The area of lake Winnembago.*"

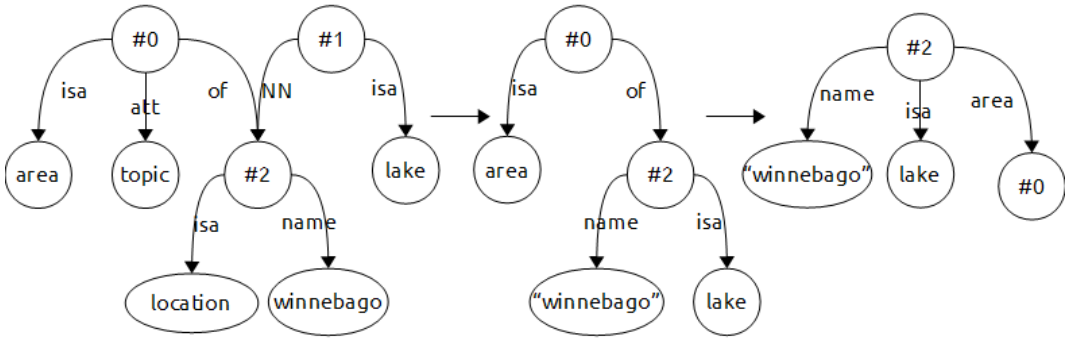


Figura 96: grafo del parsing e trasformazione della query 7

Al grafo sono stati applicati i seguenti pattern: pattern 3 sul nodo #0; pattern 4 sul nodo #2 e pattern 19 sul nodo #1; successivamente e' stato applicato il pattern 7 al nodo #0. Il grafo ottenuto e' corretto e GeX estrae con successo l'istanza richiesta, ovvero il numero 557 corrispondente all'area del lago.

Si e' voluto mettere in evidenza questa prova sperimentale perche' il grafo del parsing presenta il pattern 19 per il quale, come abbiamo esposto nel capitolo 3.4, non e' definita una risoluzione stabile e univoca. Ricordiamo che il pattern si presenta quando Boxer non riesce a rintracciare tutti gli elementi grammaticali fondamentali di una frase, come il verbo, che nel caso della nostra query e' sott'inteso e sarebbe *named*. Abbiamo applicato il pattern 19 nonostante si sia specificato precedentemente che non sempre i risultati che genera sono corretti. In questo caso specifico la trasformazione tramite il pattern e' riuscita bene e non ci sono stati errori, ma, se il grafo di Boxer presenta questo tipo di pattern si raccomanda caldamente di riformulare la query in linguaggio naturale per essere certi del buon funzionamento del software e dei pattern; il prossimo pattern e' appunto una rivisitazione della query 7 in maniera piu' dettagliata e precisa.

Query n. 8

"*L'area del lago chiamato "Winnembago".*" -> "*The area of lake named "Winnembago".*"

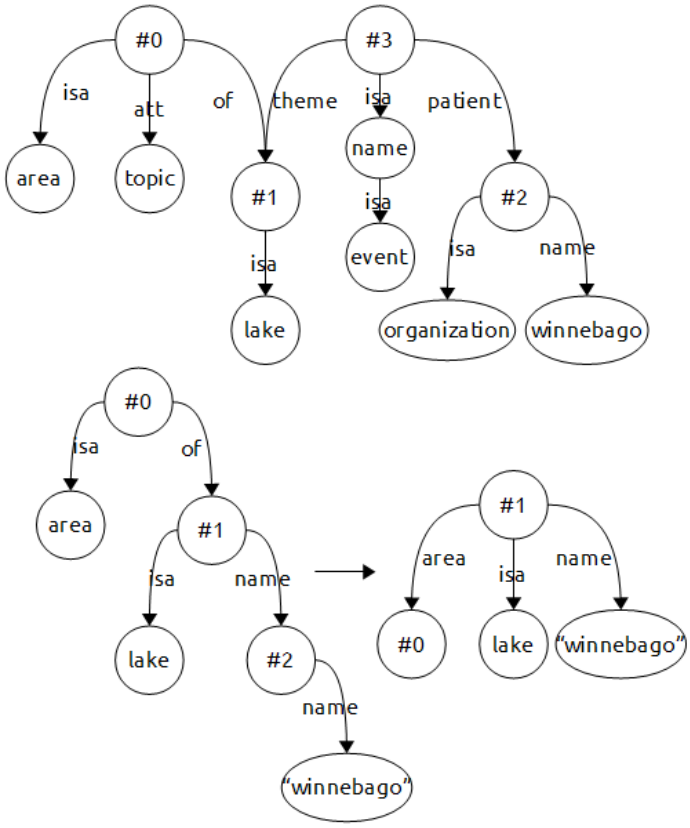


Figura 97: grafo del parsing e trasformazione della query 8

Nonostante la somiglianza tra la query 7 e la 8 notiamo che i grafi in output di Boxer sono molto diversi tra loro. Nel caso specifico della query 8 non abbiamo piu' l'arco di tipo *NN* ma abbiamo una struttura ben definita e conosciuta, il pattern 11.

Vediamo ora i pattern che sono stati utilizzati per la semplificazione del grafo: sul nodo #0 abbiamo applicato il pattern 3, sul nodo #2 il pattern 4 e sul nodo #3 il pattern 11; al secondo passaggio e' stato applicato il pattern 7 al nodo #0.

Come vediamo il grafo risultante e' assolutamente identico al grafo ottenuto della query 7, quindi anche il risultato dell'interrogazione tramite GeX e' il medesimo, ovvero l'istanza con valore 557 che indica l'area del lago.

Questa prova e' stata proposta per sottolineare come il parser generi grafi molto diversi tra loro nonostante la somiglianza che per noi possono avere due interrogazioni.

Query n. 9

"*Il fiume lungo 1638 [kilometri].*" -> "*The river length "1638".*"

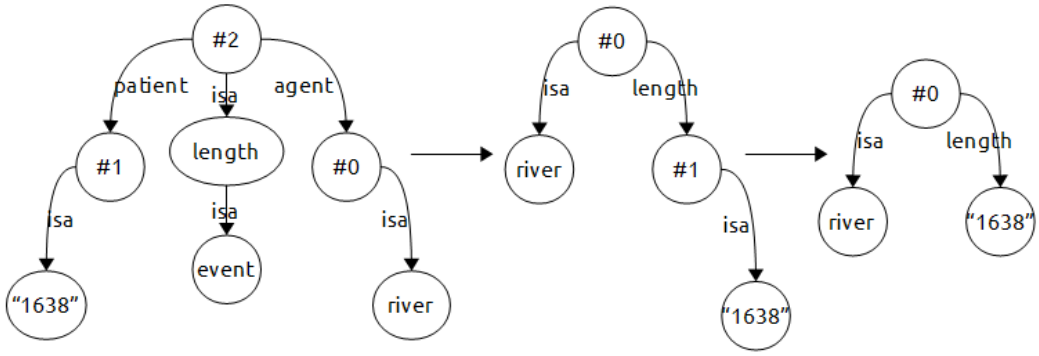


Figura 98: grafo del parsing e semplificazione della query 9

Al grafo sono applicati al primo passaggio il pattern 10 al nodo #2 e successivamente il pattern 18 al nodo #0; il grafo puo' essere sottoposo a GeX che risponde con successo all'interrogazione restituendo la stringa interessata, ovvero il fiume Red, lungo 1638 km.

Questa prova e' stata utile per testare ulteriormente alcuni pattern particolari e abbiamo verificato ulteriormente, in particolare per il pattern 18, che le approssimazioni non dipendono strettamente dal contenuto delle label (salvo casi eccezionali definiti in alcune query) e sono pertanto svincolate dal vocabolario dei dati.

4.3. Prove sperimentali su DBLP

Analizzeremo ora alcuni casi di studio di query eseguite su una collezione diversa di dati. La collezione e' stata presentata precedentemente al capitolo 2, paragrafo 2.1 e contiene le informazioni riguardanti pubblicazioni letterarie. Questa collezione, DBLP, e' stata usata esclusivamente in questa fase dello sviluppo della tesi per testare ulteriormente i pattern.

Anche per queste prove valgono le premesse esposte al paragrafo 4.1.

Query n.10

"*Quali sono le phdthesis?*" -> "*What are the phdthesis?*"

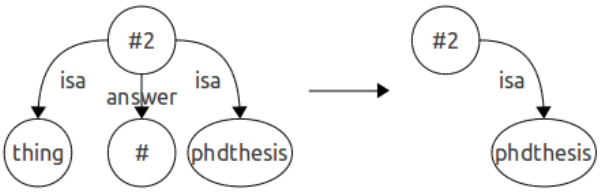


Figura 99: grafo del parsing e semplificazione della query 10

Alla query sono stati applicati i pattern 1 e 2 sul nodo #2; il grafo ottenuto non ha bisogno di modifiche e, sottoposto a GeX, ci permette di estrarre con successo tutte le produzioni letterarie del tipo specificato.

Query n. 11

"*Gli articoli dell'anno 1993.* " -> "*The articles of year "1993".*"

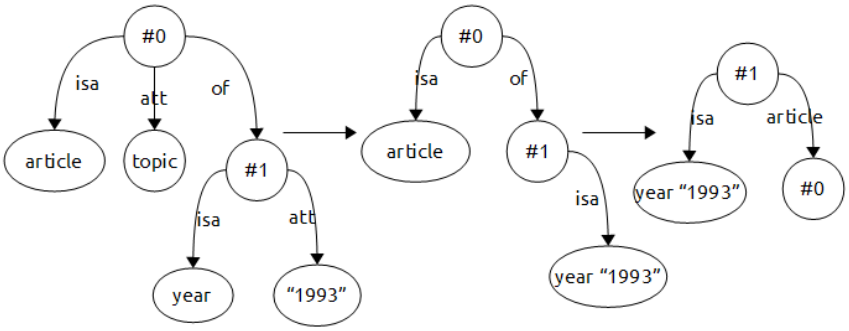


Figura 100: grafo del parsing e semplificazione della query 12

Al grafo risultante dal parsing sono stati applicati il pattern 3 al nodo #0 ed il pattern 9 al nodo #1; al secondo passaggio e' stato applicato il pattern 7 al nodo #0. Il grafo ottenuto, sottoposto a GeX, non genera alcun risultato: il software infatti non ha riscontri all'interno del database per il nodo *year "1993"* e l'arco *article*.

L'interrogazione in linguaggio naturale in esame ha senso per un interlocutore che e' a conoscenza delle strutture grammaticali che possono trovarsi nel linguaggio parlato, come ad esempio il sottintendere termini; noi infatti siamo in grado, leggendo la query, di rintracciare tutti gli elementi sottintesi e ti portare mentalmente l'interrogazione in una forma piu' precisa dove tutti gli elementi sono espressi, ad esempio la frase "*Gli articoli **scritti** nell'anno 1993*". La query 11 pero' non ha senso per uno strumento automatico come un parser grammaticale: esso infatti necessita di tutti gli elementi grammaticali fondamentali di una frase (verbo, soggetto, complemento oggetto) per effettuare una buona analisi. Ne deriva che nonostante la flessibilita' che il linguaggio naturale ha, nell'ambito di questo tipo di studio non dobbiamo forzare eccessivamente il parser fornendogli query con parti sottintes: il risultato, come in questo caso, non sarebbe soddisfacente anzi, sarebbe proprio fallimentare.

Query n. 12

Presentiamo ora una delle possibili versioni alternative della query 11, versione in cui il parser e' posizione di poter rintracciare tutti gli elementi grammaticali fondamentali e riesce a costruire un grafo corretto.

"*I libri scritti nel 1988.*" -> "*The books written in "1988".*"

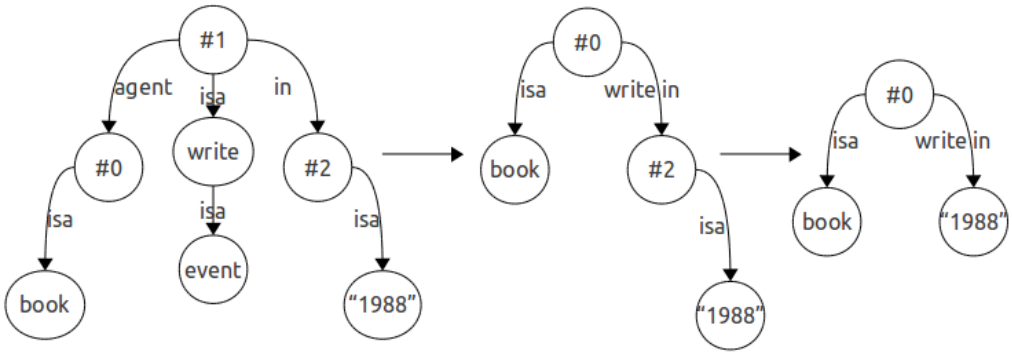


Figura 101: grafo del parsing e semplificazione della query 11

Al grafo risultante dal parsing e' stato applicato il pattern 12 al nodo #1 e successivamente abbiamo applicato il pattern 18 al nodo #0; il grafo ottenuto puo' essere sottoposto a GeX previa modifica della label "*write in*" in "*year*"; il software estrae con successo l'istanza richiesta, ovvero il titolo del testo "*Computing with Logic: Logic Programming with Prolog.*" scritto nel 1988.

Grazie a questa prova possiamo sottolineare il discorso fatto precedentemente e soprattutto mettiamo in evidenza che per la maggior parte delle query che non portano all'estrazione corretta dei dati dalla collezione interrogata esiste una query strutturata in modo diverso ma semanticamente identica che porta ad un successo.

Query n. 13

Andiamo ad introdurre una query che ci permettera' di fare importanti considerazioni.

"*Le pagine dell'entita' chiamata "Programmed Control of Asynchronous Program Interrupts.".*" - > "*The pages of the entity titled "Programmed Control of Asynchronous Program Interrupts.".*"

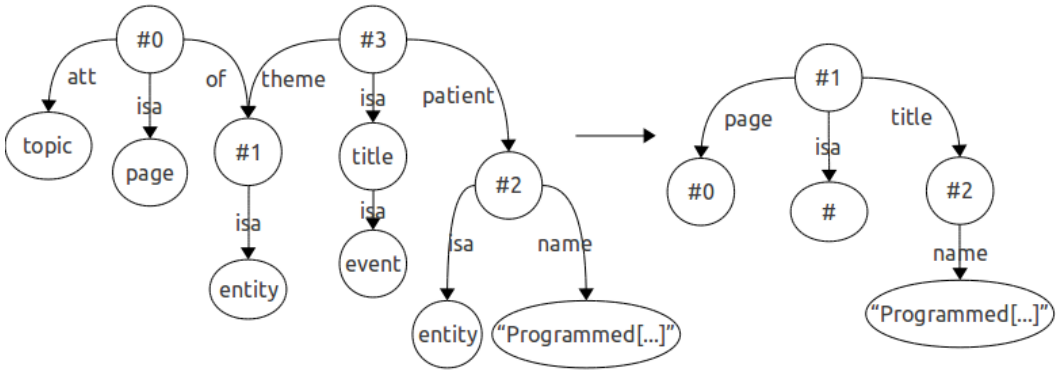


Figura 102: grafo del parsing e semplificazione della query 14

Alla query sono stati applicati il pattern 3 e 7 sul nodo #0, il pattern 4 sul nodo #2 ed il pattern 11 sul nodo #3; il grafo ottenuto utilizzato da GeX non restituisce alcuna istanza. Il motivo del fallimento risiede nel percorso (*#1, title, #2, name, "Programmed [...]"*): non esiste infatti alcun attributo *name* per i titoli delle opere, il nodo puntato della proprieta' *title* e' gia' esso stesso il valore dell'attributo. Possiamo quindi dedurre che il nodo #2 e l'arco *name* sono inutili ed e' possibile eliminarli. Tale azione corrisponde in un certo senso a quello che facciamo nell'applicazione del pattern 16 (si veda capitolo 3, paragrafo 3.3): cio' che abbiamo individuato e' quindi un nuovo pattern che vorremmo accumunare con il pattern 16 di cui presentiamo sotto una nuova versione:

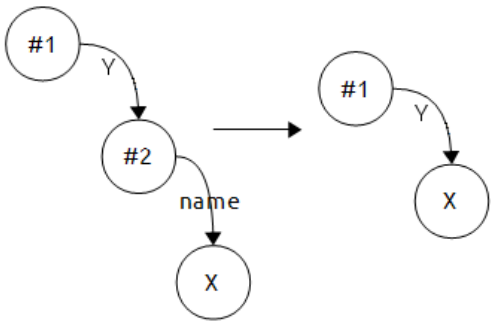


Figura 103: possibile nuova versione del pattern 16

Come si vede abbiamo introdotto un fattore di generalita' sull'arco che collega #1 e #2; tale fattore, indicato con *Y* ci permette di risolvere tutti quei casi in cui il parser, incontrando nella frase una parola con lettera maiuscola, introduce l'arco *name*. Il pattern descritto in figura pero' non e' corretto: esso infatti va anche ad approssimare situazioni in cui l'approssimazione non deve esserci come nel caso riportato sotto in un brano della figura 69 che corrisponde alla query "*What are the states that the Potomac runs through?*"

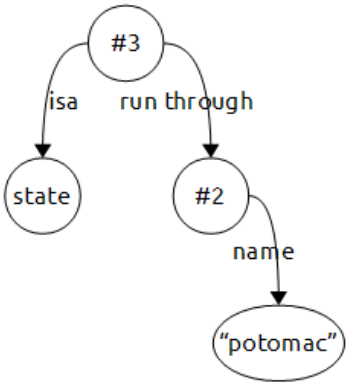


Figura 104: brano della figura 69

Se andiamo ad applicare il pattern descritto in figura 100 non otterremmo piu' risposte da parte di GeX in quanto collegheremmo direttamente l'istanza #3 al suo nome, tramite l'arco *run through*, il che non coincide con lo schema dei dati usato.

Il pattern 16 cosi' generalizzato non e' utile e non verra' applicato, ma rimane da ricercare una soluzione conveniente che possibilmente comprenda il pattern 16 e il pattern presentato in figura 99.

La query in esame ha evidenziato come l'interrogazione di insiemi di dati differenti possa portare alla scoperta di nuovi pattern; tale fatto e' dovuto alla natura stessa delle collezioni di dati. Infatti ogni collezione ha proprie strutture e vocabolari che possono portare l'utente alla formulazione di interrogazioni strutturalmente diverse e quindi a grafi di Boxer diversi che mettono in evidenza nuovi pattern.

Query n. 14

"*Gli oggetti pubblicati da Benjamin/Cummings.*" -> "*The object published by Benjamin/Cummings.*"

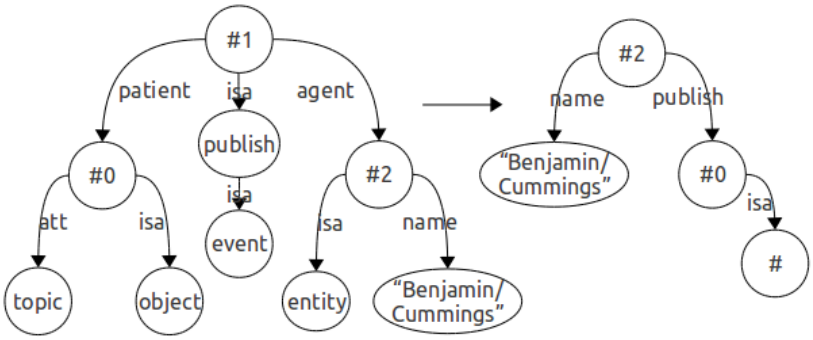


Figura 105: grafo del parsing e semplificazione della query 14

I pattern applicati sono: pattern 3 al nodo #0, pattern 10 al nodo #1 e pattern 4 al nodo #2; il grafo, previa modifica della label dell'arco "*publish*" in "*publisher*", puo' essere utilizzato da GeX ma il software non trova alcuna corrispondenza con i dati presenti nel database. Il motivo del fallimento della prova risiede nel problema gia' evidenziato nelle query 2 e 3.

Leggendo il grafo risultante dall'applicazione dei pattern abbiamo "*l'entita' chiamata "Benjamin/Cummings" pubblicata da un'entita' di tipo qualsiasi (#).*" che e' un concetto diverso da quello espresso nella nostra interrogazione iniziale. Sappiamo che l'interpretazione del significato di alcuni termini in linguaggio naturale puo' essere erroneo quindi tralasciamo per un attimo la traduzione del grafo e ne osserviamo la struttura. Come notiamo stiamo cercando un oggetto che ha un determinato nome e che ha la proprieta' *publish/publisher* connessa ad una any label: se pensiamo alla struttura della nostra collezione di dati notiamo che concretamente stiamo cercando un opera letteraria chiamata "Benjamin/Cummings"; il successo di questa query e' quindi impossibile dato che tale nome non indica il titolo di un'opera ma una casa di pubblicazione. Ecco il motivo del fallimento, ovvero lo spostamento del soggetto della nostra interrogazione dall'oggetto che ha come *publisher* Benjamin/Cummings all'entita' Benjamin/Cummings.

La soluzione di tale problema risiede come sempre nella formulazione di una query leggermente diversa dal punto di vista grammaticale e strutturale e ne vediamo subito un esempio.

Query n. 15

"*I libri che sono stati editati da [hanno come editor] Yasushi Kiyoki.*" -> "*The books which are editing by Yasushi Kiyoki.*"

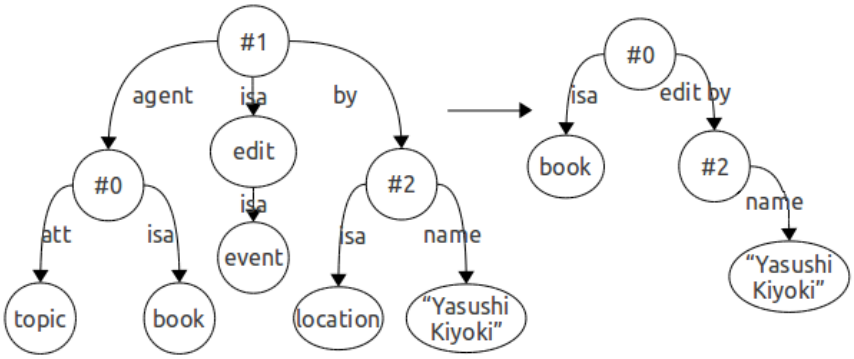


Figura 106: grafo del parsing e semplificazione della query 15

Al grafo sono stati applicati il pattern 3 al nodo #0 ed il pattern 4 al nodo #2. Come notiamo abbiamo anche applicato una variazione del pattern 10 al nodo #1 di cui parleremo a breve. Il grafo ottenuto puo' essere sottoposto a GeX previa modifica della label "*edit by*" in "*editor*"; il software estrarra' con successo l'istanza del libro editato da Yasushi Kiyoki ovvero il testo intitolato "*Information Modelling and Knowledge Bases XV, 13th European-Japanese Conference on Information Modelling and Knowledge Bases EJC 2003, Kitakyushu, Japan, June 3-6, 2003*".

Come abbiamo detto abbiamo applicato una variazione del pattern 10 in cui l'arco *patient* e' sostituito dall'arco *by*.

Tra i pattern che abbiamo rintracciato molti si fondano su una struttura in cui da un nodo (qui #1) escono tre archi di cui due sono *agent* e *isa* mentre il terzo puo' essere *patient* (pattern 10), *in* (pattern 12), *through* (pattern 17) o *by* come in questo caso.

La scoperta di questa nuova forma del pattern ci permette di generalizzare i pattern come segue:

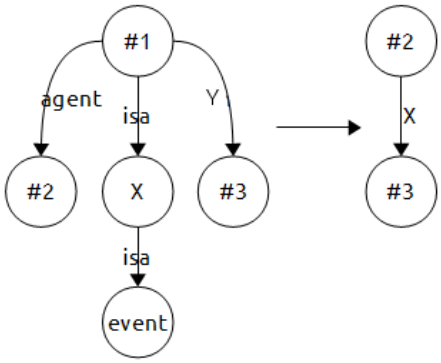


Figura 107: generalizzazione del pattern 10, 12 e 17

Abbiamo inserito un fattore di generalizzazione, la label *Y*, che ci permette di identificare con questo pattern molte strutture di pattern simili che variano solo per il contenuto della label *Y*.

Data l'elevata eterogeneita' del linguaggio naturale non e' possibile stabilire a priori un numero ben definito di patter, la loro ricerca e il loro studio deve continuare mano a mano che si interrogano nuovi dataset. Cosi' facendo l'archivio di pattern potra' essere costantemente aggiornato rimuovendo pattern obsoleti o aggiungendone di nuovi.

Query n. 16

"*Quali sono gli oggetti connessi ad una persona chiamata "Subrata Dasgupta"?*" -> "*What are the object connected to a person named "Subrata Dasgupta"?*"

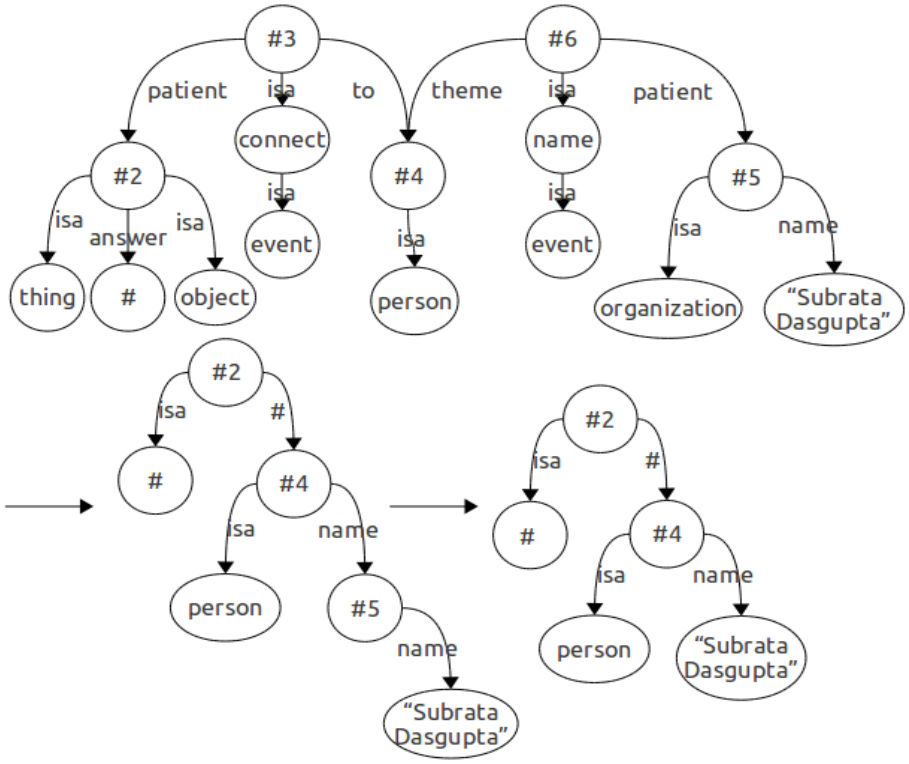


Figura 108: grafo del parsing e semplificazione della query 16

Al grafo risultante dal parsing di Boxer sono stati applicati i seguenti pattern: pattern 1 e 2 al nodo #2; pattern 4 al nodo #5; pattern 15 al nodo #3 e pattern 11 al nodo #6; successivamente e' stato applicato il pattern 16

al nodo #4. Il grafo ottenuto puo' essere utilizzato da GeX per l'interrogazione del database, interrogazione che avra' successo grazie all'estrazione delle istanze contenenti i dati dei testi a cui la persona chiamata Subrata Dasgupta e' in qualche modo connessa (solitamente come creatore).

La prova in esame e' servita per testare alcuni pattern particolari, specialmente il pattern 15 che permette di sfruttare la grande flessibilita' di GeX grazie a delle any label sugli archi.

Query n. 17

"*I titoli degli oggetti creati da una persona che ha anche creato un articolo chiamato " Statistical Processors.".*" -> "*The titles of the object created by a person who has also created an article named "Statistical Processors.".*"

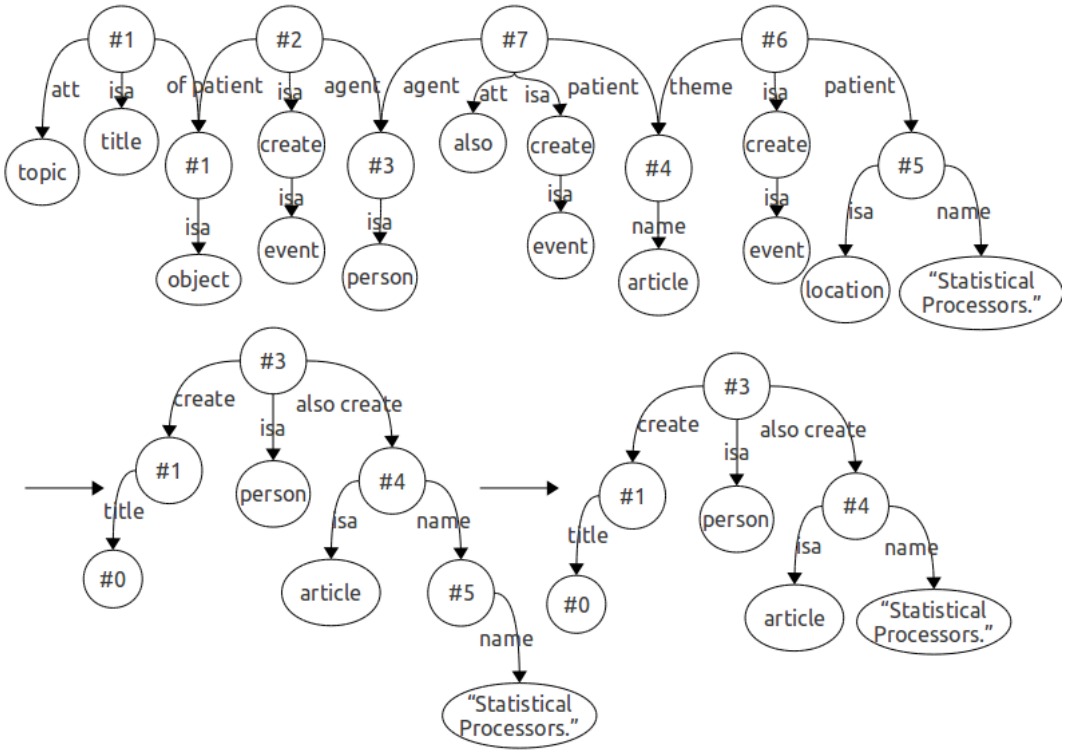


Figura 109: grafo del parsing e semplificazione della query 17

Al grafo sono stati applicati numerosi pattern: pattern 3 e 7 al nodo #0; pattern 5 al nodo #1; pattern 10 al nodo #2; pattern 4 al nodo 5; pattern 11 al nodo #6; pattern 9 e successivamente pattern 10 al nodo #7; al secondo passaggio abbiamo applicato il pattern 16 al nodo #4.

Il grafo risultante puo' essere utilizzato da GeX previa modifica delle label "*create*" e "*also create*" in "*creator*"; l'interrogazione pero' non arriva a buon fine. L'errore risiede nel verso degli archi "*create*" e "*also create*": nel grafo ottenuto dalle semplificazioni notiamo che stiamo cercando una persona che "ha creato/e' creatore" di testi ma, nello schema dei dati, non abbiamo alcuna indicazione di una struttura simile, ovvero non abbiamo una classe persona a cui e' connessa una proprieta' *creator*; in DBLP abbiamo delle opere letterarie che *sono create* da delle entita' persona, quindi idealmente il verso degli archi dovrebbe essere il seguente:

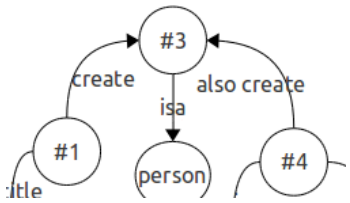


Figura 110: porzione di grafo corretto

dove le entita' #1 e #4 sono appunto le nostre opere letterarie che hanno in comune lo stesso creatore.

Provando anche a parsare una frase leggermente diversa del tipo "*The titles of the objects* ***which are*** *created by a person [...]*" il grafo risultante e' il medesimo di quello presentato in figura 106, quindi l'errore permane.

In questo caso non e' tanto la coordinazione tra i vari componenti dell'interrogazione che viene a mancare, quanto il concetto di "*creato da*", concetto espresso dalla forma grammaticale e grafica del pattern 10 (*agent-patient*) ma che sposta il soggetto della nostra frase dagli oggetti che ricerchiamo (le opere letterarie) al loro creatore. Si tratta di un problema gia' esposto precedentemente e che e' risolvibile parsando frasi

in linguaggio naturale che esprimono lo stesso concetto ma che differiscono per come vengono scritte.

Query n. 18

"*Gli oggetti creati da una persona che si chiama "Roberto Giacobazzi".*" -> "*The objects created by someone named "Roberto Giacobazzi".*"

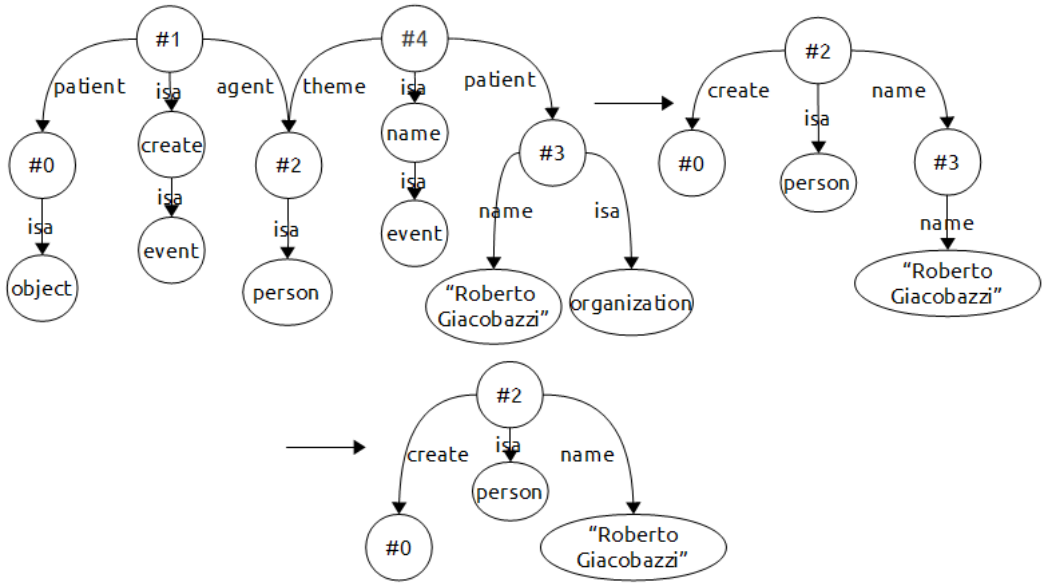


Figura 111: grafo del parsing e semplificazione della query 18

Al grafo sono stati applicati i seguenti pattern: pattern 5 al nodo #0; pattern 10 al nodo #1; pattern 4 al nodo #3 e pattern 11 al nodo #4; al secondo passaggio e' stato applicato il pattern 16 al nodo #2.

Il grafo, previa modifica della label "" in "", puo' essere sottoposto a GeX che estrae correttamente le opere di Roberto Giacobazzi, ovvero lo scritto dal titolo "*"Optimal" Collecting Semantics for Analysis in a Hierarchy of Logic Program Semantics.*".

La query presentata ha permesso di testare i pattern con strutture grammaticali (created by, edit by etc.) proprie di questi tipo di schema di dati.

4.4. Considerazioni sui risultati ottenuti

In questo paragrafo riportiamo alcune considerazioni numeriche e statistiche sulle prove sperimentali eseguite nel corso dello sviluppo della tesi.

Con il termine *successo* indichiamo le query che, sottoposte a Boxer, trasformate attraverso i pattern e sottoposte a GeX, hanno dato esito positivo estraendo i dati di interesse dalle collezioni; con *insuccesso* indichiamo le situazioni che in un qualche modo hanno portato alla mancata estrazione dei dati richiesti.

Non consideriamo i casi in cui Boxer non genera alcun grafo in output.

Osservando la totalita' delle query sperimentate possiamo affermare quanto segue.

Totale query eseguite	**93**	
Successi	62/93	**66%**
Insuccessi	31/93	**34%**

Totale query eseguite su Geobase	**72**	
Successi	49/72	**68%**
Insuccessi	23/72	**32%**

Totale query eseguite su DBLP	**21**	
Successi	13/21	**62%**
Insuccessi	8/21	**38%**

Come si vede dalle percentuali presentate i casi in cui le query in linguaggio naturale hanno portato all'estrazione corretta dei dati di interesse sono piu' della meta' della casistica osservata.

In particolar modo notiamo che la percentuale dei successi e' maggiore in relazione al database geobase rispetto che a DBLP; questo e' probabilmente dovuto al fatto che i pattern sono stati sviluppati tramite l'utilizzo di geobase e, in certi casi, possono facilitare maggiormente la semplificazione delle strutture delle query di questa collezione, mentre, per DBLP possono esistere pattern dalla struttura leggermente diversa: tali differenze possono portare quindi a fallimenti.

Presentiamo ora alcune percentuali in relazione ai fallimenti delle interrogazioni.

Motivazioni fallimenti	%
Totale query fallite: 34	
Pattern non semplificabili	**7%**
Grafi del parsing non completamente connessi	**11%**
Termini grammaticali non specificati	**24%**
Variazione del soggetto considerato	**27%**
Coordinazione tra i termini non rispettata	**31%**

Nella tabella non abbiamo inserito il mancato riconoscimento di label di archi e nodi da parte di GeX in quanto la modifica delle label viene eseguita manualmente nei casi in cui e' possibile, altrimenti solitamente la causa del fallimento risiede in un motivo piu' ampio, come lo spostamento del soggetto della query da un'entita' all'altra.

Le motivazioni dei fallimenti sono par la maggior parte riconducibili alle modalita' di costruzione di un'interrogazione in linguaggio naturale e alle limitazione che puo' presentare il parser Boxer, come, ad esempio, il non riuscire a rispettare la coordinazione tra i termini di interrogazioni complesse.

Sottolineiamo infine che per il 90% circa delle interrogazioni che falliscono e' possibile rintracciare un'interrogazione analoga ma con struttura linguistica diversa che porta ad estrarre con successo i dati.

Conclusioni

La tesi presentata e' partita dall'analisi degli standard per la gestione di tipi di dato modellati a grafo: XML e il modello RDF; successivamente siamo passati ad esporre le caratteristiche degli schemi di dati utilizzati nell'ambito della tesi e dei software utilizzati: GeX, il software per l'interrogazione flessibile ed approssimata di dati modellati a grafo, e il parser Boxer.

L'obiettivo raggiunto del lavoro svolto e' stato, in particolare, quello di individuare dei meccanismi fissi e ricorrenti che permettano il passaggio da grafi strettamente linguistici e grammaticali derivanti dal parsing di interrogazioni in linguaggio naturale, a delle query basate su grafi che permettano l'interrogazione flessibile ed approssimata di dati da collezioni, indipendentemente dal vocabolario utilizzato dai dati o dalla struttura organizzativa degli stessi.

Ognuno di questi pattern rintracciati e' stato dettagliatamente spiegato e arricchito da esempi di vario tipo.

Infine abbiamo presentato alcune tra le piu' interessanti prove sperimentali eseguite su collezioni di dati differenti, prove che hanno evidenziato le buone capacita' di utilizzo dei pattern e allo stesso tempo

hanno permesso di trarre conclusioni importanti riguardo a come porre le interrogazioni in linguaggio naturale al parser, o come trasformare correttamente i grafi.

L'ostacolo piu' difficile da superare e' stato il doversi districare tra le innumerevoli forme in linguaggio naturale che una semplice frase puo' avere; si e' osservato che non sempre e' possibile parsare e trasformare in modo corretto una frase, soprattutto a causa dell'elevatissima eterogeneita' e vaghezza che il linguaggio naturale porta con se'. Nonostante questo nella maggior parte dei casi in cui un'interrogazione non andava a buon fine venivano rintracciate una o piu' interrogazioni poste in forme linguistiche differenti ma semanticamente identiche che permettevano l'estrazione corretta dei dati dalle collezioni.

Le prove sperimentali hanno anche permesso di porre l'accento sull'importanza del continuo aggiornamento dei pattern rimuovendo dalla collezione quelli che possono essere obsoleti e aggiungendone di nuovi.

Il naturale sviluppo di questa tesi e' rappresentato dal proseguimento del progetto sull'interrogazione flessibile di dati modellati a grafo tramite interrogazioni in linguaggio naturale; in particolar modo si pensa all'implementazione di un meccanismo automatico per la trasformazione dei grafi e l'applicazione dei pattern in modo semplice e veloce ad esempio tramite un foglio di stile che sfrutti le trasformazioni XSLT o tramite la creazione di una piccola applicazione Java o Python che sfrutti il parser DOM per l'analisi e la manipolazione dei file XML.

Inoltre, come gia' accennato, sviluppo fondamentale e' il continuo aggiornamento della collezione dei pattern tramite l'utilizzo dei software e delle tecniche rintracciate su dataset sempre diversi, in modo da avere una collezione il piu' completa possibile.

Ringraziamenti

Sono giunta finalmente a questo momento.

I primi doverosi ringraziamenti vanno al prof. Martoglia sempre disponibile nell'aiutarmi e nell'indirizzarmi al meglio nel corso dello sviluppo di questa tesi; in secondo luogo devo ringraziare il prof. Johan Bos molto disponibile e che mi ha aiutata a sfruttare al meglio Boxer.

Un ringraziamento particolare va ai professori del CdL di Informatica: hanno creato un ottimo corso di studi e tutti, nel corso di questi tre anni, si sono impegnati per rendere me e i miei compagni dei buoni informatici, trasmettendoci la loro passione per questo mondo. Se posso vantare un ampio e approfondito bagaglio culturale in questo campo e' merito loro.

Un ringraziamento accorato va ai Bomber del corso di studi. In questi tre anni ho condiviso con loro le gioie e i dolori degli esami e dei pomeriggi in aula studio a Fisica, nonche' grigliate e cene davvero speciali. Tra loro ringrazio in particolare i quattro che si laureano insieme a me: Valle, Gec, Nat e Martin; una citazione speciale se la merita anche France'.

Devo ringraziare profondamente gli amici che mi hanno sempre sostenuta ed appoggiata. Gli amici vicini: Ely, Fio, Otty, Je e naturalmente Schioppo; e le amiche lontane, quelle della Femili sparse per l'Italia: Steh,

Sara, Amalia e Maria Lucia. Senza di loro non avrei potuto godere di innumerevoli momenti speciali.

Tra gli amici ringrazio anche i Maludasta, colonna sonora di quasi due anni di studio, e Debby, compagna di prove e dolcissima amica.

La mia famiglia e' stata il motore che mi ha permesso di arrivare fino a qui. Grazie ai miei genitori e a mio fratello per la fiducia riposta nelle mie capacita', per avermi spronata e per avermi spesso sopportata nei numerosi "il giorno prima dell'esame". Sono orgogliosa e felice di far parte di questa famiglia e nonostante le difficolta' sono molto grata a tutti loro per cio' che fanno ogni giorno per me.

Non posso non ringraziare Nicoletta e Marco che da quasi due anni mi accolgono e mi aprono la loro casa a Modena con grande calore e affetto.

Che altro dire... Penso sia giunto il momento del ringraziamento piu' importante.

Cio' che sono diventata come persona, come donna e come informatica e' anche merito di colui che da due anni mi sta accanto in un modo totalmente speciale: Dani-san. Egli e' in grado di donarmi serenita' anche nei momenti piu' bui e difficili, tranquillizzandomi e ricordandomi quanto sono capace. Cosa piu' importante, mi dona un affetto fuori dal comune. Ed e' per questo, e per altri milioni di motivi, che un singolo grazie non basterebbe. Mai.

Non essendomi laureata in Preveggenza non conosco il mio futuro. Ma dato che sono avvezza a pensarla secondo modelli algoritmici e matematici direi che se tutte queste persone che mi sono state vicine e che hanno creduto in me fino ad ora, permettendomi ed aiutandomi ad arrivare fin qui, continueranno a farlo, di sicuro la spinta verso il futuro sara' altamente positiva. E sono quindi altamente grata a tutti e altamente ottimista.

Grazie.

Bibliografia

[1] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)"

http://www.w3.org/TR/xml/

[2] Frank Manola, Eric Miller, "RDF Primer"

http://www.w3.org/TR/rdf-primer/

[3] Thomas H. Cormel, Charles E. Leisrson, Ronald L- Rivest, Clifford Stein, "Introduzione agli algoritmi e alle strutture dati", Seconda Edizione, McGraw-Hill, 922 - 926.

[4] F. Mandreoli, R. Martoglia, W. Penzo, "Approximating Expressive Queries on Graph-modeled Data: the GeX Approach", in pubblicazione, 2010.

[5] F. Mandreoli, R. Martoglia, W. Penzo, G. Villani, "Flexible Query Answering on Graph-modeled Data", in: EDBT, 2009.

[6] "C & C tools"

http://svn.ask.it.usyd.edu.au/trac/candc/

[7] James R. Curran, Stephen Clark, Johan Bos, "Linguistically Motivated Large-Scale NLP with C&C and Boxer", 33 - 36, 2009.

http://aclweb.org/anthology-new/P/P07/P07-2009.pdf

[8] Johan Bos, "Wide-Coverage Semantic Analysis with Boxer", 277 - 286, 2008.

http://www.meaningfactory.com/bos/pubs/Bos2008STEP2.pdf