Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Polo Territoriale di Como

Corso di Laurea Specialistica in Ingegneria Informatica

Raccomandazione di visualizzazioni e percorsi esplorativi per risultati di ricerche multi-dominio su Web

Relatore: Prof. Marco Brambilla
Correlatore: Prof. Alessandro Bozzon

Tesi di laurea di:

Mattia Berlusconi Matr. 754720
Andrea Mauri Matr. 754721

Anno Accademico 2011-2012

Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Polo Territoriale di Como

Master of Science in Computer Engineering

Visualizations and exploration paths suggestion for multi-domain results of Web searches

Supervisor: Prof. Marco Brambilla
Assistant supervisor: Prof. Alessandro Bozzon

Master Graduation Thesis by:

Mattia Berlusconi Student ID 754720
Andrea Mauri Student ID 754721

Academic Year 2011-2012

Sommario

Oggi i motori di ricerca sono diventati molto potenti e forniscono risultati costituiti da grandi volumi di dati. L'utente, interagendo con questi sistemi, deve riuscire ad esplorare correttamente i dati in modo da estrarre informazioni significative che soddisfino le necessità della sua ricerca. Questa operazione però, non è banale quando si ha a che fare con una grande quantità di dati.
Lo scopo di questo lavoro di tesi è quello di studiare una soluzione per assistere l'utente suggerendogli un modo efficiente per esplorare i dati.
In questo documento proponiamo due algoritmi: il primo ha lo scopo di proporre all'utente la visualizzazione più adatta dei risultati della ricerca, mentre il secondo mira a generare una serie di diverse visualizzazioni collegate tra loro che diano una visione più ampia sui risultati.
Per raggiungere il primo obiettivo vengono fatte due tipologie di analisi: la prima sui tipi dei dati mentre la seconda sulle distribuzioni delle istanze risultate dalla particolare ricerca.
Per il secondo obiettivo è stato creato un modello a grafo che rappresenta lo spazio delle rappresentazioni ed un algoritmo per la sua esplorazione.
Questi algoritmi sono stati implementati all'interno di un prototipo e sono stati testati coinvolgendo anche persone estranee alle strategie adottate.

Abstract

Nowadays search engines are very efficient and provide results composed by huge data volumes. The user has to interact with these systems, through which he can explore the data in order to extract significative information. This activity it's not trivial if you have to deal with a huge amount of data. The aim of this thesis is to study a method that can assist the user, suggesting a smart way to explore the data.
Two algorithms has been developed: the first one with the aim to deliver to the user the most fitting view according to the query results, the second, instead, aims to create an sorted sequence of views in order to give to the user a wider sight of the results.
To achieve the first goal, two kind of analysis are performed: the first looks at the data type, while the second analyzes the distribution of the result instances.
To achieve the second goal, we created a graph model, that represents the views space, and we developed an algorithm to explore it.
Eventually these algorithms were implemented within a prototype and validated by means of a internal and external test.

Indice

Elenco delle figure

Introduzione

Contesto e motivazioni

Fin dall'avvento del Web uno dei problemi fondamentali dell'utente è stato quello di ricercare informazioni. L'attività di ricerca, negli anni si è molto evoluta.
Oggi i dati disponibili sul Web diventano sempre più numerosi e complessi. Questo aumento di complessità dei dati ha portato i sistemi di ricerca ad evolversi molto rapidamente per effettuare ricerche sempre più sofisticate.
Di pari passo con l'aumento della complessità dei dati, aumentano anche le esigenze degli utenti. Essi vogliono sistemi di ricerca intelligenti, che siano in grado di capire le proprie esigenze, espresse sotto forma di query molto complesse. Si aspettano anche risultati sempre più precisi, ben organizzati e conformi alle proprie attese.
Oggi i sistemi di ricerca più avanzati permettono di eseguire quelle che vengono chiamate ricerche multi-dominio, ricercano dati attraverso più sorgenti, ognuna specializzata in un particolare dominio, poi li integrano e li presentano all'utente finale.
Questi sistemi però restituiscono come output all'utente volumi di dati sempre più grandi e complessi che sono difficili da esplorare.
L'utente deve essere quindi assistito durante l'esplorazione dei risultati di una ricerca: da solo non è in grado di estrarre dai dati informazioni significative.

Obiettivi

L'obiettivo di questa tesi è quello studiare una soluzione che permetta di assistere l'utente durante l'esplorazione dei dati che vengono restituiti da un sistema di ricerca Web multi-dominio.
Il metodo che si vuole proporre analizza i dati forniti in output da un motore di ricerca e in base alle loro caratteristiche (sia per quanto riguarda il tipo di dati che per la distribuzione delle istanze) propone all'utente una modalità di visualizzazione adatta ed efficace.
In particolare abbiamo strutturato il lavoro in due fasi successive:

- la prima prevede la creazione di un algoritmo che mostri all'utente le migliori visualizzazioni dei risultati ottenuti dalla sua ricerca;

- la seconda, più avanzata, prevede che venga proposto all'utente un "percorso" seguendo il quale egli possa esplorare i dati attraverso una serie di diverse visualizzazioni in sequenza collegate tra loro da un filo logico. In altre parole in questa seconda fase si vuole proporre all'utente una serie di visualizzazioni diverse sui dati nell'ordine più appropriato.

Infine gli algoritmi sono stati validati coinvolgendo persone che non erano a conoscenza delle strategie utilizzate per progettare gli algoritmi.

Struttura della tesi

Qui di seguito illustriamo la struttura di questa tesi:

Capitolo 1 Background, in questa sezione viene presentato l'ambito nel quale il lavoro che abbiamo svolto si colloca. Vengono poi presentate alcune ricerche svolte in passato correlate con gli argomenti trattati in questa tesi.

Capitolo 2 Approccio, qui viene descritta dettagliatamente la strategia che è stata adottata per raggiungere gli obiettivi di questa tesi. Vengono presentati gli algoritmi che sono stati sviluppato ad hoc per questo progetto.

Capitolo 3 Implementazione ed esperimenti, in questa parte di tesi viene descritta l'implementazione degli algoritmi che sono stati presentati nel capitolo precedente, poi vengono descritti i test che sono stati effettuati per validarli.

Capitolo 4 Conclusioni e sviluppi futuri; vengono fatte alcune considerazioni finali e descritte alcune idee per sviluppi futuri.

Bibliografia Vengono riportati in questa sezione tutti i riferimenti dei testi ed articoli scientifici consultati durante la realizzazione del progetto e durante la stesura di questa tesi.

Capitolo 1

Background

Il lavoro che viene presentato in questa tesi si colloca all'interno del progetto SeCo. In particolare in questo capitolo presenteremo, in un primo momento le caratteristiche generali di questo progetto, poi descriveremo dettagliatamente dove si colloca, all'interno del SeCo, il nostro lavoro.

1.1 Il progetto SeCo

Il progetto SeCo[1] (sigla di Search Computing) è un progetto finanziato dalla Comunità Europea a cui prende parte il Politecnico di Milano e si pone come obiettivo quello di sviluppare un motore di ricerca che permetta all'utente di creare ed eseguire query complesse, sfruttando molteplici servizi Web come sorgenti di dati.
Definiamo innanzi tutto cosa si intende per query multi-dominio: è una query attraverso la quale l'utente si aspetta una ricerca su più domini diversi, ognuno di essi specializzato nella raccolta di dati appartenenti ad un certo contesto.
Facciamo un esempio molto semplice: un utente vuole visitare un museo a Roma e vuole cercare un ristorante vicino per pranzare.
Fatta una ricerca del genere, l'utente si aspetterà di vedere i musei di Ro-

[1] http://www.search-computing.it/

ma e relativi ristoranti ordinati secondo alcuni parametri che possono essere: la vicinanza, il prezzo medio, la categoria, ecc... Da questo concetto nasce il termine "multi-dominio": l'utente vuole avere informazioni riguardo più concetti diversi (in questo particolare caso Musei e Ristoranti).

Descriviamo brevemente il funzionamento e le parti essenziali dell'architettura del SeCo [1]. Per fare ciò utilizziamo la figura 1.1 che illustra l'architettura generale e i principali moduli del sistema SeCo.

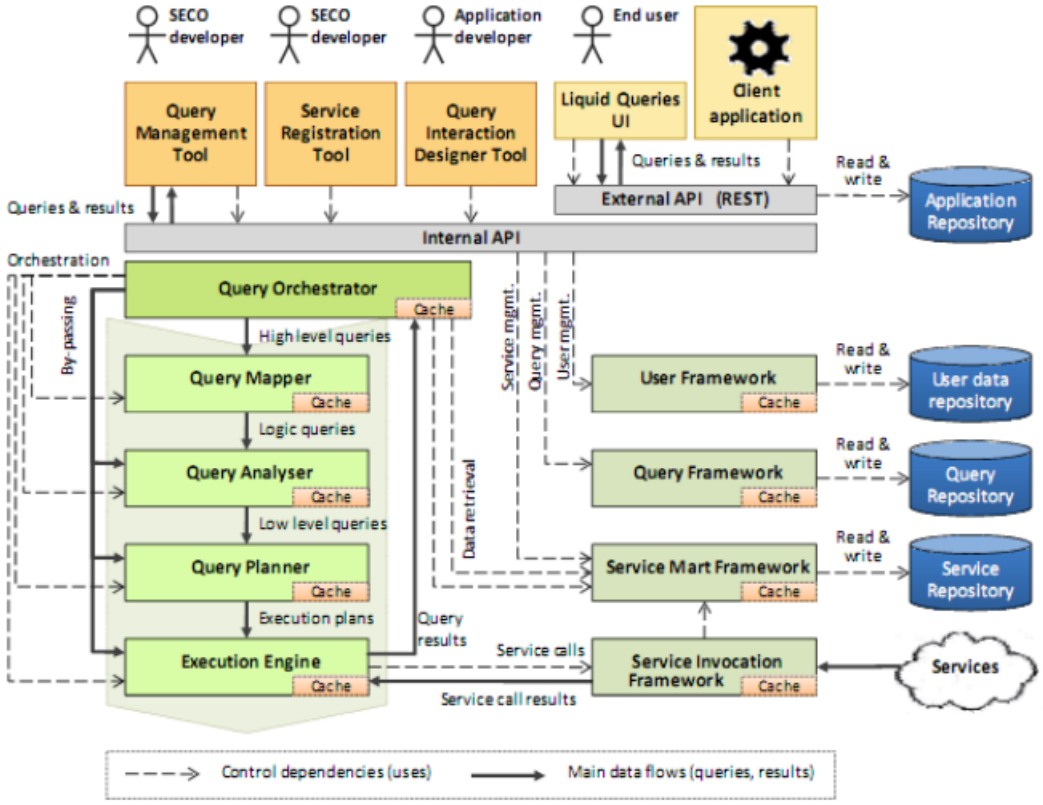

Figura 1.1: Architettura del framework SeCo

Per prima cosa il modulo "Orchestrator" si occupa di scomporre la query in più parti in base al dominio di appartenenza. A questo punto il "Query Mapper" associa ognuna delle sottoquery che vengono generate alla sorgente di dati più appropriata per ogni particolare dominio di interesse (in questo

particolare esempio vengono considerate le sorgenti dati dei musei e dei ristoranti). Le sorgenti dei dati per essere utilizzate dal SeCo devono essere registrate attraverso una particolare procedura che verrà descritta in seguito. Ogni sotto-query viene quindi passata al "Query Analyzer" che si occupa di tradurla in un formato interno a basso livello studiato appositamente per rendere la successiva fase di ottimizzazione più efficace.

A questo punto le query ottimizzate vengono passate al componente "Query Planner" che produce un piano di esecuzione che indica la precisa sequenza di passi da seguire per l'esecuzione della query.
Infine il piano di esecuzione viene concretamente eseguito dal "Execution Engine" il quale si occupa di:

- effettuare le chiamate ai vari servizi designati, per le singole ricerche, dal "Query Mapper" attraverso il Service Invocation Framework. Chiamiamo servizi le varie sorgenti dei dati all'interno delle quali vengono effettuate le ricerche nei singoli domini;
- creare i risultati delle query combinando gli output prodotti dalle chiamate ai servizi;
- calcolare il ranking globale dei risultati della query;
- riportare i risultati ordinati rispetto al loro rank globale;

Quella appena descritta è l'architettura del framework, la quale può essere utilizzata e personalizzata dai programmatori secondo le loro esigenze per costruire particolari applicazioni.

Vediamo ora brevemente come possono essere sviluppate delle applicazioni sulla base del framework SeCo. E' interessante descrivere questo processo in quanto successivamente, nella sezione di questa tesi (capitolo 2) in cui descriveremo il lavoro che abbiamo svolto faremo spesso riferimento a termini citati in questa fase.
Definiamo in particolare le tre tipologie di utenti che sono coinvolti in questo processo:

Publishers Sono coloro che si occupano di rendere accessibile da parte del sistema SeCo una particolare sorgente di dati. Essi devono fare in modo che una particolare sorgente di dati sia compatibile all'interfaccia prevista dal sistema per la registrazione dei servizi.

Expert Users Questa categoria identifica coloro che creano delle applicazioni che si basano sul framework messo a disposizione dal SeCo. Questi utenti selezionano i "Service Mart" a cui sono interessati e li "connettono" semanticamente tra loro creando dei "connection pattern". I "Service Mart" rappresentano concettualmente l'insieme di tutti servizi Web relativi ad un certo dominio utilizzati dal SeCo. Essi identificano quindi il concetto di entità (per esempio Museo, Ristorante o Albergo). I "connection pattern" rappresentano invece un'astrazione delle relazioni tra i "Service Mart"; essi vengono quindi costruiti per rappresentare una relazione semantica tra due oggetti appartenenti a due diversi domini [8].

End Users Questi sono gli utenti che si limitano semplicemente ad utilizzare le applicazioni create dagli "Expert Users". Essi interagiscono con le applicazioni creando delle query e visualizzando i relativi risultati forniti dal sistema. A seconda dei risultati che vengono visualizzati, essi rifiniscono meglio le query che sottopongono al sistema ed esplorano i risultati da vari punti di vista interagendo con le interfacce di visualizzazione dei dati.

Descriviamo ora i passi fondamentali per la creazione di un'applicazione basata sul framework SeCo.

Sviluppo di un servizio di ricerca In questa fase gli utenti "Publisher" sviluppano dei servizi Web che rendono disponibile un certo volume di dati. In realtà questa fase viene eseguita solo nel caso in cui i dati richiesti dall'applicazione non siano reperibili da nessuna sorgente esistente sul Web.

Registrazione di un servizio Questa fase è necessaria qualora un "Expert User" voglia registrare al sistema una sorgente dati già presente sul

Web ma ancora non utilizzata dal SeCo. Tali sorgenti esistono nel Web ma necessitano di essere adattate per essere utilizzate dal Search Computing Framework. La loro interfaccia deve soddisfare dei requisiti che sono definiti attraverso alcuni template standard.

Configurazione dell'applicazione Questa è la fase in cui un programmatore sviluppa una propria applicazione che utilizza il framework del SeCo. Innanzi tutto bisogna scegliere quali sorgenti dati si vogliono utilizzare, bisogna definire le loro configurazioni e i "connection pattern".
In particolare poi bisogna impostare alcuni parametri per configurare le modalità di visualizzazione dei dati forniti dalla ricerca. Quest'ultima operazione è molto importante in relazione lavoro che abbiamo svolto. Nella parte di questa tesi in cui descriveremo il nostro lavoro spiegheremo più dettagliatamente come permetteremo al programmatore di definire questi parametri e come terremo conto di essi per presentare all'utente i dati generati dalla ricerca.

Raffinamento del piano di esecuzione della query Questo passo consiste nello specificare manualmente un procedimento fortemente ottimizzato per la costruzione del piano di esecuzione delle query.

1.1.1 Liquid query

Liquid query è un paradigma con il quale l'utente finale effettua ricerche attraverso una qualsiasi applicazione basata sul framework del SeCo [2].
Questo è un approccio basato sulla ricerca esplorativa delle informazioni e permette all'utente di manipolare, estendere e raffinare una query in base ai risultati ottenuti.
Il SeCo offre una serie di primitive attraverso le quali l'utente può modificare la composizione e la visualizzazione dei risultati della ricerca.
In particolare l'utente, attraverso queste primitive, ha la possibilità di espandere la query aggiungendo nuovi servizi per la ricerca, richiedere più risultati da uno specifico servizio, aggiungere o rimuovere degli attributi, aggregare i

risultati oppure ordinarli.
Di particolare interesse per il nostro lavoro sono le primitive per la visualizzazione dei dati con le quali l'utente può scegliere la rappresentazione che preferisce.
Le Liquid Query sono caratterizzate da un proprio ciclo di vita:

- "l'Expert User" configura un template della query per la specifica applicazione;
- la query specificata dall'utente finale viene inviata al SeCo;
- la query viene eseguita dal sistema e viene generato il result-set;
- l'utente esplora i risultati ottenuti

Il nostro lavoro è strettamente connesso all'ultima fase del ciclo di vita.

1.2 Collocazione nel progetto SeCo

Questo paragrafo specifica dove, all'interno del progetto SeCo, prende posto il lavoro che abbiamo svolto.

La soluzione che abbiamo creato si occupa di analizzare i risultati della ricerca restituiti dal SeCo e genera la migliore visualizzazione per mostrarli all'utente finale.
La visualizzazione migliore deve mostrare all'utente le caratteristiche principali dei dati mettendo in evidenza le entità fondamentali, ma deve anche evidenziare peculiari caratteristiche del particolare result-set preso in considerazione. Tali caratteristiche dipendono non solo dalla struttura e dal tipo di dati selezionati ma soprattutto dalla distribuzione dei valori delle istanze degli oggetti che compongono i risultati ottenuti da una particolare ricerca effettuata attraverso il sistema.
A seconda della distribuzione dei valori infatti potrebbe essere significativo mostrare o meno un certo attributo oppure mostrare i suoi valori in funzione di un certo altro attributo. Potrebbe accadere che i valori un attributo abbiano una certa distribuzione che rendano quest'ultimo molto interessante, ma

l'utente potrebbe non accorgersene in quanto osserva i dati da una diversa prospettiva.

Attualmente il SeCo dispone già di un client che, attraverso un'interfaccia grafica, mostra i dati all'utente. Questa interfaccia però deve essere configurata per intero manualmente dall'utente finale. Egli deve configurare molte impostazioni, in particolare deve decidere quali oggetti visualizzare, quali specifici attributi mostrare o meno e con quale livello di dettaglio.
Viene inoltre lasciata all'utente la scelta di quale widget di visualizzazione utilizzare per mostrare i dati (mappa, grafico a dispersione, grafico a barre, time-line o lista) e su quale dimensione (o asse) della rappresentazione mettere ogni singolo attributo. Tutti questi widget di visualizzazione esistono già nel client del SeCo.
Potremmo pensare che tutte queste impostazioni disponibili e che il fatto di lasciare all'utente la facoltà di fare tutte queste scelte riguardo le modalità di visualizzazione dei dati sia un aspetto positivo.
In realtà non lo è per le ragioni espresse nell'introduzione: l'utente si troverebbe davanti ad una grande massa di dati molto complessi e non sarebbe in grado, a colpo d'occhio, di fare un'analisi accurata per decidere quale sia la visualizzazione migliore.
Un'implementazione della nostra soluzione prenderebbe posto fisicamente sul server, e nel workflow delle operazioni svolte, si inserirebbe alla fine di tutte le attività svolte dall'engine. Praticamente avrebbe il compito di fornire al client, ed in particolare ai widget di visualizzazione, informazioni su come i dati risultati dalla ricerca debbano essere presentati all'utente.

La soluzione proposta si vuole quindi sostituire all'utente nel processo di configurazione delle impostazioni dell'interfaccia grafica.
L'utente potrà così accedere direttamente alla modalità di visualizzazione più adatta in base al tipo dei dati ed ai valori che gli attributi assumono nella ricerca che ha effettuato.
All'utente verrà quindi automaticamente proposta una serie di visualizzazioni per i dati.

1.3 Related work

In questa sezione riportiamo alcune ricerche che abbiamo trovato in letteratura nell'ambito del nostro lavoro.

1.3.1 Tassonomie delle visualizzazioni

Abbiamo trovato due ricerche che hanno come obiettivo quello di classificare le visualizzazioni di un certo insieme di dati.
Rodrigues et al [4] hanno analizzato le tecniche di creazione di una visualizzazione e le hanno scomposte in due parti principali: la spazializzazione e lo sfruttamento dello stimolo visuale "pre-attentive".
La prima consiste in un procedimento di trasformazione dei dati grezzi in un formato spaziale più adatto ad essere percepito, mentre la seconda parte consiste nell'identificare come una serie di elementi visuali (forma, colore, dimensione, posizione) possano essere sfruttati per massimizzare l'efficenza il potere espressivo di un'interfaccia grafica.
Sono stati identificati diversi tipi di spazializzazione come per esempio la "structure exposition" nella quale viene sfruttata la struttura intrinseca dei dati per crearne una rappresentazione ad alto livello (ad esempio le amicizie su Facebook possono essere visualizzate attraverso un grafo).
Per quanto riguarda la seconda parte della creazione della concreta visualizzazione, sono state identificate diverse modalità di utilizzo degli stimoli "pre-attentive", per esempio il colore può essere utilizzato per veicolare una sensazione di differenza o corrispondenza tra due o più dati.
In conclusione a questa ricerca, gli autori, mostrano che sono riusciti a classificare varie tecniche di visualizzazione utilizzando gli elementi da loro definiti. In particolare hanno elencato, per ogni tipo di visualizzazione (mappa, diagramma a torta, diagramma a barre, ecc...), una serie di tecniche da utilizzare per la spazializzazione e per massimizzare lo sfruttamento dello stimolo "pre-attentive".

Una seconda ricerca [5] introduce il modello di "stato dei dati", attraverso il quale utilizza due categorie di elementi per descrivere una data tecnica

di visualizzazione: un insieme di passi per la trasformazione dei dati e un secondo insieme di tipi di operazioni che si possono effettuare sui dati.
Il primo insieme è composto da quattro elementi:

- **Value**: rappresenta il dato grezzo;
- **Analytical Abstraction**: rappresenta informazioni aggiuntive che descrivono i dati, vengono chiamati anche meta-dati;
- **Visualization Abstraction**: rappresenta le informazioni che possono essere visualizzate riguardanti i dati;
- **View**: è la visualizzazione finale che l'utente può osservare e interpretare;

Il secondo è composto da tre tipi di trasformazioni:

- **Data Transformation**: sono trasformazioni che estraggono alcune informazioni aggiuntive sui dati;
- **Visualization Transformation**: questa categoria di trasformazioni prende come input i meta-dati e genera dei dati in un formato più adatto per la visualizzazione;
- **Visual Mapping Transformation**: questa famiglia di trasformazioni genera una visualizzazione partendo dai dati creati dalla trasformazione precedente.

Questa ricerca è prettamente teorica e ha lo scopo di creare una tassonomia delle visualizzazioni sui dati, ma può essere utile anche a agli sviluppatori perchè permette di osservare alcuni metodi consolidati per creare nuove visualizzazioni.

Questa ricerca, insieme alla precedente, è utile per quanto riguarda la scelta delle visualizzazioni date le caratteristiche dei dati che vogliamo mostrare. Entrambe infatti forniscono delle metodologie per mappare una certa struttura dati con la visualizzazione più adatta.

.

1.3.2 Sistemi di generazione automatica di visualizzazioni

Alcuni esempi di sistemi automatici per la visualizzazione dei dati si possono trovare nell'ambito dei database.
Per esempio con il progetto DARE [6] Tiziana Catarci et al hanno sviluppato un sistema che si basa fondamentalmente sull'uso dei predicati. Vengono definite alcune regole attraverso le quali si descrivono le caratteristiche dei dati e viene scelta una visualizzazione.
In particolare in questo progetto vengono utilizzate quattro categorie di predicati:

- **regole visuali**: definiscono i simboli visuali da utilizzare e le loro caratteristiche (ad esempio i simboli linea e punto e le loro caratteristiche come colore, dimensione forma);
- **regole dei dati**: definiscono le caratteristiche del modello dei dati;
- **regole di mappatura**: stabiliscono quale attributo è adatto ad essere visualizzato attraverso quale elemento grafico;
- **regole di percezione**: definiscono quanto ha senso visualizzare un certo attributo attraverso un certo simbolo.

Il sistema che è stato sviluppato basandosi sulle regole appena elencate, dato il result-set generato dalla ricerca effeuttata dall'utente, riesce a trovare la migliore rappresentazione dei dati considerati.

Un secondo esempio di sistemi di questo tipo è il progetto Polaris [7], un sistema per interrogazione, analisi e visualizzazione nell'ambito di database multidimensionali. La generazione di una visualizzazione definita in questo progetto è composta da tre passi:

- **configurazione delle visualizzazioni**: si definisce quali dati devono essere mostrati su quali dimensioni in una data visualizzazione;

- **identificazione del tipo di grafico**: questa operazione viene svolta automaticamente dal sistema analizzando i tipi di dati presenti sugli assi;
- **impostazione proprietà grafiche dell'interfaccia**: in questa fase vengono assegnate le caratteristiche visuali del marker quali forma, orientamento, grandezza e colore.

Una volta eseguite le fasi dell'algoritmo appena elencate, viene mostrata all'utente la visualizzazione più adatta dei dati.
Anche una volta generata la visualizzazione, l'utente può configurare in tempo reale le sue proprietà, può scegliere di visualizzare nuovi attributi o cambiare le caratteristiche visuali della rappresentazione.

Le ricerche che abbiamo analizzato basano i metodi che propongono esclusivamente sull'analisi dei tipi primitivi dei dati (interi, booleani, ecc...), solo il progetto DARE [6] introduce alcuni concetti più astratti per diversificare i tipi di dati utilizzando informazioni "semantiche".
Questi concetti ci possono essere utili per quanto riguarda la prima fase dell'analisi dei dati, che si concentra sul modello dei dati.
D'altra parte però, nessuna delle ricerche esaminate effettua un'analisi delle istanze dei dati che si vogliono visualizzare. Riteniamo che quest'ultimo tipo di analisi sia molto utile per il nostro lavoro in quanto osservando la distribuzione dei valori che assumono i dati è possibile creare una visualizzazione più flessibile e adatta per ogni singolo insieme di dati considerato. Istanze diverse degli stessi dati potrebbero richiedere diversi tipi di visualizzazione per meglio mostrare le loro caratteristiche.

1.3.3 Generazione automatica di un percorso di visualizzazione

La seconda parte del nostro lavoro si inserisce nell'ambito delle "ricerche esplorative".
Questo tipo di ricerca richiede uno sforzo maggiore sia da parte dell'utente, sia da parte del motore di ricerca che deve fornire un interfaccia che sia

in grado di presentare i dati in maniera tale che possano essere facilmente esplorati.
Il termine "ricerca esplorativa" fu introdotto da Marchionini [10] e comprende tutte quelle attività che possono essere classificate come "investigative" o "di apprendimento", differenziandole dalle attività di semplice ricerca ("lookup").
Per permettere questo tipo esplorazione, i motori di ricerca devono offrire all'utente un'interfaccia grafica interattiva che gli permetta di esplorare i risultati in maniera intuitiva ed efficiente.
Ad esempio Kosmix [9] è un motore di ricerca ed esplorazione di argomenti. Questo software offre all'utente tutte le informazioni riguardo un dato argomento riassunte in una pagina e permette, data una query creata dall'utente, di trovare tutti i temi correlati.
Un altro esempio è dato da SenseMaker [11], un'interfaccia per l'esplorazione delle infromazioni provenienti da sorgenti eterogenee. Questo sistema propone come risposta alla query dell'utente i risultati dei servizi utilizzati per la ricerca in una tabella, raggruppati per sorgente.
A questo punto l'utente può esplorare il result-set in tre modi: può decidere di aggiungere o togliere dalla visualizzazione uno o più attributi, può decidere di approfondire la ricerca selezionando una o più particolari sorgenti da cui estrarre più dati rispetto a quelli già mostrati, oppure ha la possibilità di raffinare la query per avere risultati più specifici.
Gli ultimi due progetti presentati prongono diverse modalità per facilitare all'utente l'attività di esplorazione di dati. Questi metodi non offrono però degli strumenti automatici: essi richiedono infatti una forte interazione dell'utente per cambiare il punto di vista della presentazione dei dati.

Una ricerca che molto si avvicina a quello che è l'obiettivo della seconda parte del nostro lavoro è stata fatta da alcuni studenti del Politecnico di Milano [12].
Questi studenti hanno studiato la possibilità di applicare nell'ambito della ricerca di informazioni, algoritmi utilizzati in robotica per l'esplorazione di spazi.

Lo spazio di esplorazione può essere rappresentato con un grafo in cui i nodi rappresentano le varie sorgenti di informazione interrogabili per una possibile ricerca mentre gli archi indicano se l'output del nodo di partenza (result-set della prima ricerca) può essere utilizzato come chiave per interrogare la sorgente rappresentata dal nodo di arrivo.
In questo modo viene creato un "percorso" di navigazione utilizzando due possibili algoritmi di esplorazione del grafo: in profondità e in ampiezza.
Allo stato attuale di questo progetto è richiesta, durante la creazione del percorso, un'interazione dell'utente: egli deve infatti confermare ogni scelta che l'algoritmo prende per selezionare uno degli archi uscenti da un nodo. Se l'utente non conferma una scelta, l'algoritmo propone un arco diverso.

Capitolo 2

Approccio

In questo capitolo vengono illustrati gli algoritmi che sono stati ideati per perseguire gli obiettivi di questa tesi.

2.1 Selezione della miglior visualizzazione dei dati

Il primo obiettivo di questa tesi è quello di sviluppare un algoritmo che sia in grado di proporre automaticamente la migliore visualizzazione possibile per i dati ottenuti dalla ricerca. La scelta della visualizzazione dei dati dovrà essere fatta analizzando le caratteristiche del result-set e del result-model ottenuti.

Questo algoritmo prende come input un result-set ed il relativo result-model. Un result-set consiste in un insieme di tuple contenenti i risultati della ricerca multi-dominio effettuata dall'utente. Questi dati sono ottenuti dall'engine del SeCo che si è occupato di aggregare i dati provenienti dai diversi domini sui quali è stata effettuata la ricerca.
Ogni oggetto del result-set proviene da un certo servizio ed è costituito da attributi che possono essere dei vari tipi: integer, double, string, boolean, ecc.
Un result-model è invece uno schema che esprime informazioni riguardo il tipo e le principali caratteristiche di tutti gli attributi di ogni singolo oggetto

presente nella tupla.
Il nostro algoritmo si occupa di analizzare il result-set considerando le proprietà del modello che vengono inferite dal result-model. Il fine ultimo di questa analisi è quello di individuare la migliore visualizzazione per rappresentare i risultati della ricerca.
Definiamo cosa si intende, nell'ambito di questo lavoro, per "visualizzazione". Essa è un modello concettuale che ci permette di rappresentare dati combinati tra loro, è costituita da sette dimensioni, su ognuna delle quali possono essere rappresentati i valori di un attributo degli oggetti che compongono i dati che si vogliono visualizzare. Le sette dimensioni di una generica visualizzazione sono le seguenti:

- Asse X
- Asse Y
- Asse T
- Clue color
- Clue size
- Clue shape
- Clue info.

Le prime due dimensioni rappresentano i due assi di un classico diagramma cartesiano, la dimensione T è un asse che è adatto per rappresentare valori di attributi temporali, mentre le altre quattro dimensioni rappresentano le possibili caratteristiche dei marker dei dati. Anch'esse possono essere utilizzate per veicolare delle informazioni.
Innanzi tutto vogliamo determinare quali attributi sono adatti per essere rappresentati su ogni singolo asse della visualizzazione. In seguito all'analisi dei dati potrebbe anche accadere che una dimensione non venga associata a nessun attributo.
In base al tipo degli attributi che verranno assegnati agli assi della visualizzazione, verrà poi scelto il tipo di widget grafico da utilizzare per presentare i

dati all'utente. Tale widget grafico si occupa di rappresentare concretamente la visualizzazione e può essere di vari tipi: mappa, grafico cartesiano, grafico a barre, lista o timeline.

Descriviamo ora i passi dell'algoritmo necessari per selezionare la migliore visualizzazione dei dati. Nella seguente figura troviamo una rappresentazione grafica del flusso di lavoro.

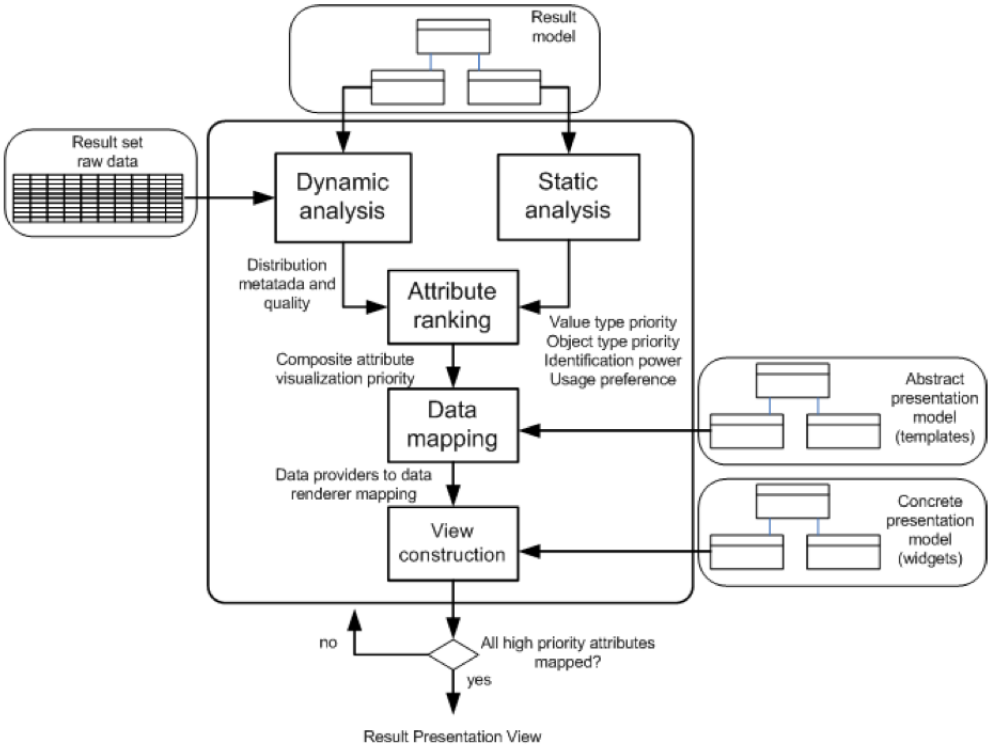

Figura 2.1: Flusso di lavoro dell'algoritmo di selezione delle visualizzazioni

Come si può vedere dalla figura 2.1, ci sono cinque principali passi che portano alla selezione della visualizzazione:

- **Analisi statica**: valuta le caratteristiche proprie del tipo degli attributi;

- **Analisi dinamica**: valuta le caratteristiche della distribuzione delle istanze dei dati;
- **Ranking attributi**: assegna un punteggio ad ogni attributo;
- **Mapping dei dati**: assegna ad ogni dimensione della visualizzazione un attributo;
- **Mapping sui tipi di widget**: per ogni visualizzazione generata seleziona il tipo di widget più appropriato.

Qui di seguito questi passi vengono descritti in dettaglio.

2.1.1 Analisi statica

Questa parte dell'algoritmo si occupa di estrarre informazioni utili dal result-model. Queste informazioni sono associate staticamente ad ogni singolo attributo degli oggetti coinvolti dalla ricerca e ci aiutano nel processo di scelta della visualizzazione, esse possono essere suddivise in due principali categorie: informazioni dipendenti dai dati e informazioni dipendenti dall'applicazione. Nel primo caso le informazioni riguardano esclusivamente il tipo degli attributi. Elenchiamole e descriviamole:

- **Rank correlation**: assume valori tra 0 e 1. Indica quanto l'attributo considerato sia utile per il calcolo del ranking totale delle tuple ottenute nel result-set;
- **Usage preference**: assume valori tra 0 e 1. Ad ogni attributo vengono associati sette valori diversi di usage preference, uno per ogni dimensione delle visualizzazioni. Ogni singolo valore indica quanto l'attributo considerato è adatto ad essere visualizzato sulla data dimensione della visualizzazione;
- **Identification power**: assume valori tra 0 e 1. Indica quanto l'attributo considerato identifichi un'istanza dell'oggetto a cui esso appartiene.

Tali valori devono essere configurati dallo sviluppatore dell'applicazione. Per quanto riguarda le informazioni del secondo tipo (dipendenti dall'applicazione che si vuole realizzare) esse sono personalizzabili a seconda delle esigenze di chi sviluppa ("Expert User") un'applicazione che sfrutta il framework messo a disposizione dal SeCo. Tali informazioni sono:

- **Object type priority**: assume valori tra 0 e 1. Indica quanto è importante mostrare, in una visualizzazione, attributi dell'oggetto a cui tale valore è associato;
- **Attribute type priority**: assume valori tra 0 e 1. Indica quanto è importante mostrare, in una visualizzazione, l'attributo a cui tale valore è associato.

Abbiamo definito tali informazioni "dipendenti dall'applicazione" in quanto attributi od oggetti della stessa natura possono essere molto importanti per una certa applicazione e meno per altre. Le due informazioni appena elencate servono appunto a configurare queste differenze.

Durante l'implementazione abbiamo notato che il numero di tutte queste informazioni (dipendenti dai dati e dipendenti dall'applicazione) necessarie per l'analisi statica era molto grande: bisogna definire 11 valori per ogni attributo di ogni oggetto appartenente ai domini coinvolti dalla ricerca ogni volta che si vuole realizzare un'applicazione che utilizzi il framework offerto dal SeCo.
Abbiamo quindi deciso di definire alcuni "tipi semantici". Essi hanno il compito di raggruppare i vari attributi al fine di ridurre il numero di valori da definire per creare un'applicazione. In questo modo, per ogni applicazione, bisogna definire gli 11 valori appena citati solo per ogni tipo semantico e non per ogni attributo.
In particolare abbiamo definito i seguenti tipi di dato all'interno dei quali, a nostro parere, possono essere raggruppati molti attributi:

- **Indirizzo**: identifica tutti quegli attributi, solitamente di tipo stringa, che rappresentano un indirizzo postale;

- **Nome**: identifica tutti quegli attributi, solitamente di tipo stringa, che rappresentano il nome di una certa entità;

- **Statistica**: identifica tutti quegli attributi, solitamente di tipo numerico, che rappresentano una statistica riguardo qualcosa. Per fare alcuni esempi potremmo citare per uno Stato il numero di nascite e di morti, oppure per quanto riguarda un'università la percentuale di ammissione o ancora per quanto riguarda un avvocato la percentuale di cause vinte.

- **Rank**: identifica tutti quegli attributi, di tipo numerico, che sono candidati ad essere utilizzati come riferimento per ordinare le tuple del result-set.

- **Categorico**: identifica tutti quegli attributi che possono essere utilizzati per suddividere le tuple in più categorie. Ad esempio, il numero di stelle di un albergo è un attributo categorico in quanto può assumere un numero limitato di valori ben definiti. In questo esempio le tuple possono quindi essere suddivise in insiemi distinti a seconda del numero di stelle.

- **Id**: in questo tipo semantico ricadono tutti quegli attributi che sono utilizzati come informazioni identificative dei singoli oggetti contenuti nelle tuple;

- **Longitudine**: semplicemente identifica tutti quegli attributi numerici che esprimono un valore della coordinata longitudine del GPS;

- **Latitudine**: in modo duale al precedente questo tipo semantico identifica tutti quegli attributi numerici che esprimono un valore della coordinata latitudine del GPS;

- **Campo calcolato**: rappresenta tutti quegli attributi che vengono calcolati dal modulo del SeCo che effettua la query facendo delle operazioni matematiche tra due attributi di due oggetti diversi tra quelli presenti nelle tuple del result-set;

- **Info**: rappresenta i valori di quegli attributi che sono utili da mostrare come informazioni riguardo un certo oggetto, ma che non avrebbe senso utilizzare come attributi di riferimento per ordinare le tuple o per mostrare i dati in funzione di essi;

- **Distanza**: è un caso particolare di campo numerico o campo calcolato; esprime specificatamente la distanza tra due punti geografici. Può essere calcolata tra due punti geografici riportati in due distinti oggetti di una tupla, oppure può essere riportata direttamente all'interno di un attributo (per esempio un albergo potrebbe avere un attributo che indica la distanza dalla stazione più vicina). Secondo noi vale la pena di specificare un tipo apposito per questa categoria di dati in quanto rappresenta un caso diffuso in molti scenari di ricerca e va trattato in modo diverso dagli altri campi calcolati;

- **Prezzo**: è un particolare tipo numerico che non può essere ricondotto a nessuno dei tipi precedentemente descritti in quanto esprime una caratteristica particolare e ben definita di un oggetto. Pertanto richiede che venga trattato separatamente: ha delle esigenze particolari per quanto concerne la visualizzazione.

Raggruppando in questo modo i possibili tipi di dato permettiamo allo sviluppatore di applicazioni di poter definire i valori dell'analisi statica solo per questo limitato numero di "tipi semantici".
Nel caso in cui un particolare attributo di una sorgente dati che lo sviluppatore di un'applicazione volesse utilizzare non dovesse ricadere in nessuna delle categorie sopra elencate è comunque possibile associare ad esso una particolare configurazione ad hoc per i valori dell'analisi statica.

2.1.2 Analisi dinamica

Questa parte dell'algoritmo ha l'obiettivo di arricchire le informazioni ottenute attraverso l'analisi statica con le caratteristiche delle istanze dei dati del result-set considerato. In particolare vogliamo estrarre informazioni di natura statistica dal result-set.

In questa fase vengono effettuate diverse analisi sui valori dei dati ottenuti dalla ricerca che l'utente ha effettuato per capire, a prescindere dall'importanza (dettata dall'analisi statica) del particolare attributo o oggetto, se abbia senso visualizzare su una certa dimensione delle visualizzazioni un certo attributo o meno.
Nell'analisi dinamica appunto, per fare questo tipo di valutazioni, si tiene conto delle caratteristiche della distribuzione dei valori degli attributi delle istanze prese in considerazione. Potrebbe accadere che abbia senso mostrare un dato attributo in una particolare ricerca, e abbia meno senso mostrare lo stesso attributo in una seconda ricerca che pur coinvolge gli stessi oggetti aventi gli stessi attributi. Questo avviene perché tra una ricerca ed un'altra sugli stessi domini cambiano le istanze degli oggetti che vengono selezionate dall'engine e restituite nel result-set. I valori per un certo attributo potrebbero quindi essere caratterizzati da una diversa distribuzione.
L'analisi dinamica genera delle informazioni di natura statistica che vengono calcolate per ogni singolo attributo degli oggetti presenti nel result-set generato dalla ricerca multi-dominio. Prima di calcolare tali informazioni, per ogni attributo rimuoviamo le istanze duplicate degli oggetti a cui essi appartengono. Rimuoviamo le istanze duplicate dei valori per ogni singolo attributo solo per quanto riguarda il calcolo delle informazioni dell'analisi dinamica perché altrimenti lo stesso valore verrebbe analizzato più volte. Nel result-set, per ogni singolo oggetto, sono presenti delle istanze duplicate perché esso è il risultato di una combinazione di tutte le istanze dei vari oggetti (oggetti che compongono il result-model) ottenute dalle ricerche nei vari domini. Le informazioni statistiche che calcoliamo sono le seguenti:

- **Range**: è la dinamica dei valori di un attributo e viene calcolata come la differenza tra il valore massimo e il valore minimo ottenuti per un certo attributo;

- **Risoluzione**: è un valore numerico che viene calcolato come:

$$r = \frac{range}{n} \tag{2.1}$$

dove n è il numero di valori dell'attributo senza duplicati.
La risoluzione rappresenta la distanza ideale dalla quale dovrebbero essere distanziati due valori di un attributo di due istanze diverse per essere rappresentati su di un'interfaccia grafica in modo che siano ben distinguibili tra loro. Bisogna sottolineare che ovviamente la risoluzione è definibile e viene calcolata solo ed esclusivamente per gli attributi di tipo numerico. Il concetto di risoluzione verrà chiarito meglio in seguito quando utilizzeremo concretamente questa informazione nell'algoritmo;

- **Unclustered distribution quality**: può assumere valori compresi tra 0 e 1. E' un parametro che indica il livello di qualità della distribuzione dei valori di un certo attributo. Un attributo i cui valori sono uniformemente distribuiti all'interno della sua dinamica avrà un alto valore del parametro "Unclustered distribution quality";

- **Clustered distribution quality**: può assumere valori compresi tra 0 e 1. E' un parametro che ci indica quanto abbia senso mostrare i risultati ottenuti dalla ricerca raggruppati secondo i valori dell'attributo considerato.

Spieghiamo dettagliatamente come vengono concretamente calcolati i parametri appena citati. I procedimenti che verranno descritti qui di seguito vengono svolti per ogni singolo attributo del result-set. Per quanto riguarda i primi due parametri il procedimento consiste nella banale applicazione delle rispettive formule. Descriviamo invece nel dettaglio i procedimenti per il calcolo degli ultimi due parametri.

Unclustered distribution quality

Consideriamo in input i soli valori dell'attributo considerato ordinati in ordine crescente. Innanzi tutto dobbiamo dire che distinguiamo due diversi sotto-procedimenti: uno per gli attributi numerici e uno per gli attributi non numerici.

Caso attributi numerici Per prima cosa calcoliamo la differenza tra i valori presenti presi a coppie in ordine crescente per quanto riguarda l'at-

tributo considerato. Confrontiamo ora ogni valore delle differenze ottenute con il valore di risoluzione. Se due valori di un attributo hanno tra loro una distanza inferiore alla risoluzione vorrà dire che ha poco senso rappresentarli singolarmente in quanto in una ipotetica mappa, grafico o qualsivoglia visualizzazione finale, essi verrebbero rappresentati sovrapposti l'uno con l'altro e risulterebbero quindi indistinguibili.
In seguito calcoliamo media e varianza sui valori delle differenze ottenute. Possiamo ora calcolare matematicamente il parametro di Unclustered distribution quality applicando la seguente euristica:

$$Q_u = \frac{\frac{\sum_i^n (d_i - r)}{n}}{\sigma} \tag{2.2}$$

Dove d_i è la *i-esima* differenza tra valori consecutivi considerata, n è il numero di differenze, r è la risoluzione calcolata nella formula (2.1) e σ è la deviazione standard delle differenze.
Secondo l'euristica (2.2) un attributo i cui valori tendono ad essere distribuiti uniformemente avrà un valore della "Unclustered distribution quality" alto.

Caso attributi non numerici Per quanto riguarda gli attributi non numerici dobbiamo utilizzare un metodo diverso in quanto non possiamo applicare alcuna formula matematica ai valori.
Abbiamo quindi adottato la seguente strategia: per prima cosa contiamo i valori distinti che sono rimasti dopo la rimozione dei duplicati. Poi dividiamo il numero calcolato per il numero totale delle tuple che compongono il risultato della ricerca. Il quoziente di questa ultima divisione rappresenta il valore della "Unclustered distribution quality" per quanto riguarda il caso non numerico.

$$Q_u = \frac{n_{distinct}}{n} \tag{2.3}$$

Un attributo non numerico avrà quindi un valore alto di questo parametro se presenta molti valori distinti.

Dopo aver calcolato "l'Unclustered distribution quality" per ogni singolo attributo tutti i valori ottenuti vengono normalizzati rispetto al valore massimo

per avere valori che siano tutti compresi tra 0 e 1. In questo modo possiamo confrontare tra loro i valori ottenuti.

Clustered distribution quality

Anche in questo caso esistono due metodi differenziati rispettivamente per il calcolo del parametro riferito agli attributi numerici e agli attributi non numerici.

Caso attributi numerici Per il caso degli attributi di tipo numerico inizialmente costruiamo i gruppi (cluster) contenenti i relativi valori. Abbiamo definito che uno o più valori appartengono ad un gruppo se la loro differenza è minore della risoluzione calcolata precedentemente. Quindi confrontiamo tra loro i valori (che sono ordinati) e li raggruppiamo se la loro differenza è minore della risoluzione. In questo modo raggruppiamo tutti i valori che, rappresentati su di un'interfaccia, risulterebbero sovrapposti. Contiamo i gruppi ottenuti e ne terremo conto alla fine nell'euristica che useremo per il calcolo del valore finale della "Clustered distribution quality".
Ora calcoliamo altri due parametri utili per l'euristica: media e varianza delle cardinalità di ogni gruppo ottenuto.
Infine combiniamo tutti questi valori nell'euristica (2.4) per ottenere il valore finale del parametro "Clustered distribution quality":

$$Q_c = \frac{\frac{\sum_i^K c_i}{K}}{\sigma^2 + K} \tag{2.4}$$

Dove K è il numero dei gruppi trovati, i è la cardinalità dell'*i-esimo* gruppo e σ^2 è la varianza delle cardinalità dei gruppi.
Un attributo avrà "Clustered distribution quality" alta se presenta valori raggruppabili in gruppi le cui cardinalità sono omogenee ed ampie. La media definita al numeratore premia le distribuzioni con gruppi numerosi, mentre il denominatore premia le distribuzioni composte da gruppi aventi cardinalità omogenee.

Caso attributi non numerici Il procedimento utilizzato per il calcolo della "Clustered distribution quality" per gli attributi non numerici è del tutto uguale a quello per gli attributi numerici fatta eccezione per il fatto che due o più valori di un certo attributo appartengono allo stesso gruppo solo se sono uguali tra loro. Questo perché, per i nostri fini, non avrebbe senso definire il concetto di similitudine per le stringhe.

Dopo aver calcolato la "Clustered distribution quality" per ogni singolo attributo tutti i valori ottenuti vengono normalizzati rispetto al massimo per avere valori che siano tutti compresi tra 0 e 1.

2.1.3 Ranking attributi

Il ranking degli attributi ha come obiettivo quello di redarre una sorta di "classifica" degli attributi candidati ad essere visualizzati sulla data dimensione di una visualizzazione. Per ogni dimensione esiste una particolare euristica che viene utilizzata per calcolare una sorta di punteggio. Le prime tre dimensioni condividono la stessa euristica. Questi punteggi sono utilizzati per redigere una classifica attraverso la quale vengono scelti gli attributi da assegnare, per una determinata visualizzazione. Descriviamo dettagliatamente il procedimento adottato per il calcolo dei punteggi.

Assi X, Y, T L'euristica utilizzata per il calcolo del punteggio di un attributo per queste dimensioni della visualizzazione è:

$$P_{x,y,t} = usage * (objectTypePriority + attributeTypePriority * quality) \tag{2.5}$$

dove *usage* varia per ognuna delle tre dimensioni trattate e corrisponde al parametro *usage preference* specifico della dimensione X, Y o T ricavato dai dati dell'analisi statica. Anche i due successivi parametri della formula sono ricavati direttamente dai dati dell'analisi statica e non hanno bisogno di ulteriori chiarificazioni, mentre l'ultimo parametro è il valore massimo tra i due "Unclustered distribution quality" e "Clustered distribution quality" calcolati nell'analisi dinamica.

I parametri *usage*, *objectTypePriority* e *attributeTypePriority* sono stati inseriti in questa euristica in quanto vogliamo tenere conto rispettivamente di quanto l'attributo considerato è adatto ad essere visualizzato nella visualizzazione sulla dimensione considerata e quanto è importante mostrare il singolo attributo e l'oggetto a cui appartiene.
E' stato scelto di utilizzare la maggiore tra le due quality perché abbiamo bisogno di tenere conto, nell'euristica, di un parametro che valuti la qualità della distribuzione dei valori di un certo attributo e, se stiamo trattando un attributo i cui valori hanno una distribuzione che suggerisce una rappresentazione dei dati a gruppi (clusterizzata) la qualità sarà meglio rappresentata dal valore "Clustered distribution quality" altrimenti ha più senso utilizzare la "Unclustered distribution quality";

Clue size L'euristica utilizzata per il calcolo del punteggio di un attributo per questa dimensione della visualizzazione è:

$$p_{size} = usage*(objectPriority+attributePriority*rankCorrelation*quality) \tag{2.6}$$

I primi tre e l'ultimo parametro vengono ricavati in maniera del tutto analoga al caso precedente per gli assi X, Y e T (temporale). L'unica differenza sta nel fatto che ovviamente ora lo *usage* si riferisce al parametro *usage preference* della dimensione clue size, inoltre abbiamo un nuovo parametro, *rankCorrelation*, ma anch'esso viene semplicemente estratto dai dati dell'analisi statica. In questa euristica abbiamo introdotto il parametro *rankCorrelation* perché vogliamo che gli attributi che vengono rappresentati attraverso le caratteristiche fisiche del marker siano utili a valutare in termini quantitativi la tupla considerata rispetto alle altre.

Clue shape L'euristica utilizzata per il calcolo del punteggio di un attributo per questa dimensione della visualizzazione è:

$$P_{size} = usage * (objectPriority + attributePriority * quality) \tag{2.7}$$

In questo caso l'euristica che viene adottata sembra del tutto uguale a quella delle prime tre dimensioni. Ma la fondamentale differenza sta nel calcolo del parametro *quality*. Infatti non corrisponde più al massimo tra le due quality, gli viene invece direttamente assegnato il valore della "Clustered distrubution quality". Questo perché la forma (shape) del marker può rappresentare solo pochi valori discreti quindi vogliamo che vengano più favorevolmente assegnati a questa dimensione gli attributi i cui valori hanno una distribuzione adatta ad essere clusterizzata.
Per fare alcuni esempi possiamo dire che questa dimensione è adatta per rappresentare un attributo che indica il numero di stelle di un albergo, mentre non è affatto indicata per rappresentare il nome dell'albergo o la latitudine delle sue coordinate GPS.

Clue color Per questa dimensione vale lo stesso discorso fatto per la precedente e anche l'euristica utilizzata è la stessa. Infatti anche i colori del marker possono assumere solo pochi valori discreti e sono pertanto adatti a rappresentare attributi i cui valori hanno una distribuzione le cui caratteristiche fanno sì che siano adatti ad essere rappresentati a gruppi.

Clue info In questo caso la formula è:

$$P_{info} = usage * (objectTypePriority + attributeTypePriority * quality) \tag{2.8}$$

e la *quality* è, come nei primi casi il valore massimo tra la "Clustered distributed quality" e la "Unclustered distributed quality" perché la dimensione info è adatta a visualizzare attributi i cui valori possono essere sia raggruppabili che non.

2.1.4 Mapping dei dati

Questa parte dell'algoritmo si occupa di analizzare le "classifiche" redatte in precedenza al fine di generare una o più visualizzazioni per la rappresentazione dei dati presenti nel result-set. La generazione di una singola visualizzazione avviene assegnando ad ogni dimensione un attributo dei dati

del result-set.
Descriviamo dettagliatamente come vengono scelti gli attributi da associare ad ogni dimensione di una singola visualizzazione.

Come abbiamo detto nel paragrafo precedente c'è una parte dell'algoritmo che si occupa di creare una classifica di attributi candidati per essere visualizzati su ogni singola dimensione.
La parte di algoritmo che stiamo descrivendo ora riceve appunto in input queste classifiche e il suo obiettivo è quello di assegnare ad ogni dimensione della visualizzazione un attributo. La strategia che abbiamo adottato è quella di assegnare ad ogni singola dimensione l'attributo avente il punteggio più alto nella classifica della dimensione considerata. Ovviamente nel caso in cui un attributo dovesse avere il più alto punteggio in due o più classifiche riferite a due o più diverse dimensioni della visualizzazione esso non viene ripetuto ma viene associato ad una sola dimensione.
Bisogna però necessariamente fare alcune considerazioni su questa strategia: le visualizzazioni che questa soluzione genera hanno sempre senso? Ogni attributo può essere mostrato in funzione di qualsiasi altro attributo appartenente alla stessa tupla in una singola visualizzazione?
Per capire meglio qual'è il problema che si vuole sollevare facciamo un esempio concreto: un utente effettua una ricerca tra musei, ristoranti e alberghi nella città di Roma. Ipotizziamo, una volta calcolati i rank e di conseguenza le classifiche degli attributi per ogni dimensione della visualizzazione, di avere selezionato per gli assi X e Y rispettivamente latitudine e longitudine dei musei di Roma e per la forma (shape) dei marker il numero di stelle che indicano la qualità degli alberghi di Roma trovati dal motore di ricerca.
La domanda che ci poniamo è: avrebbe senso una simile visualizzazione? La risposta è no. Visti gli attributi che sono stati assegnati alle prime due dimensioni X e Y i marker indicheranno la posizione geografica dei musei di Roma. Stando a ciò che è risultato dall'assegnazione degli attributi alle dimensioni le forme dei vari marker dovrebbero dipendere dal numero di stelle dell'albergo. Ma quale albergo? Ogni marker a quale albergo si riferisce? Non abbiamo, e nemmeno esiste, una relazione 1 a 1 tra albergo e museo.

Siamo quindi giunti alla seguente conclusione: in una singola visualizzazione ha senso mostrare solo attributi appartenenti ad un unico oggetto oppure attributi calcolati sul singolo oggetto. Oppure ancora possiamo mostrare dei campi che riportano informazioni ricavate da attributi appartenenti ad altri oggetti purché siano calcolati per aggregazione. Per ricondurci all'esempio di alberghi, musei e ristoranti, possiamo dire che attraverso il marker di un museo possiamo mostrare il prezzo medio degli alberghi che si trovano nel suo intorno. Continuerebbe invece a non avere senso mostrare l'attributo nome dell'oggetto albergo in quanto su tale attributo non possiamo fare nessun tipo di aggregazione.
Abbiamo definito che una visualizzazione che abbia le caratteristiche appena elencate si dice omogenea.
L'unico caso in cui avrebbe senso mostrare attributi non aggregati appartenenti ad altri oggetti sarebbe quello in cui l'utente abbia effettuato una selezione in precedenza: per esempio nel caso di musei, alberghi e ristoranti, l'utente potrebbe aver selezionato un particolare museo; nella visualizzazione successiva avrebbe senso mostrare un attributo del museo selezionato anche se essa mostra le caratteristiche degli alberghi.

A questo punto l'algoritmo è in grado di proporre all'utente la migliore visualizzazione dei dati ottenuti dalla ricerca che ha effettuato. Solitamente però l'utente medio non si accontenta di una singola visualizzazione, ma vuole vedere i risultati della ricerca da più punti di vista.
Vogliamo quindi trovare un modo di proporre all'utente più di una visualizzazione. Analizzando quest'ultimo requisito siamo giunti alla seguente conclusione: l'algoritmo deve generare due diversi insiemi di visualizzazioni: visualizzazioni innestate e visualizzazioni alternative. I due insiemi si differenziano tra loro per le modalità attraverso le quali le visualizzazioni che contengono sono collegate tra loro.

Visualizzazioni alternative

Le visualizzazioni alternative (figura 2.2) rappresentano una serie di diversi punti di vista indipendenti tra loro ma tutti rappresentanti lo stesso result-

set.

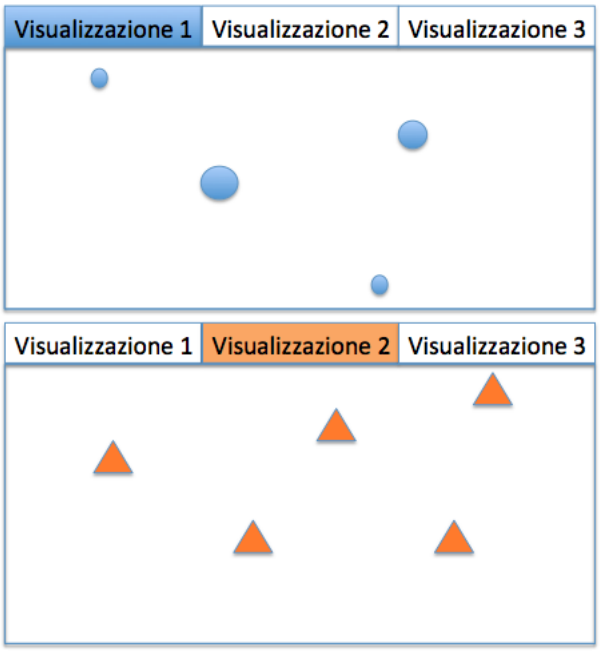

Figura 2.2: Visualizzazioni alternative

Il procedimento adottato per generare questo tipo di visualizzazione è il seguente: la prima visualizzazione viene scelta con il metodo illustrato all'inizio di questo capitolo, poi per trovarne altre, si prendono gli attributi che nelle varie classifiche delle dimensioni delle visualizzazioni stanno nelle posizioni immediatamente successive agli attributi già utilizzati. Mano a mano che si creano nuove visualizzazioni il rank degli attributi utilizzati scende e di pari passo diminuisce anche la significatività delle visualizzazioni. All'utente verranno proposte solo quelle più significative.
In questo caso tutte le visualizzazioni alternative che vengono proposte devono essere omogenee per tutte le motivazioni illustrate quando abbiamo introdotto la definizione di visualizzazione omogenea: l'utente nel caso delle visualizzazioni alternative non fa alcuna scelta riguardo le istanze degli oggetti da visualizzare. Quindi, durante la concreta costruzione di ogni singola visualizzazione dobbiamo controllare che tutti gli attributi che vogliamo

mostrare appartengano allo stesso oggetto oppure siano dei campi calcolati anche su attributi di oggetti diversi ma per aggregazione.

Visualizzazioni innestate

Le visualizzazioni innestate sono delle rappresentazioni (figura 2.3) dei dati che l'utente esplorerà una per volta, in maniera consecutiva. All'utente viene proposta quella che secondo il nostro algoritmo è la visualizzazione migliore, successivamente può decidere di approfondire la ricerca cambiando il punto di vista ed escludendo alcune tuple del result-set a cui non è interessato. Chiarifichiamo meglio la situazione riprendendo il precedente esempio di alberghi musei e ristoranti: la visualizzazione migliore che viene generata dall'algoritmo mostra tutti i musei di Roma, a questo punto se l'utente seleziona uno dei musei probabilmente si aspetterà di vedere tutti gli alberghi o ristoranti di Roma nell'intorno del museo selezionato. Da questo ragionamento nasce appunto il termine "visualizzazioni innestate".
Vediamo come concretamente generiamo queste "visualizzazioni innestate".

Figura 2.3: Visualizzazioni innestate

Come primo passo mostriamo all'utente la migliore visualizzazione ottenuta con l'algoritmo illustrato all'inizio di questo capitolo. Successivamente, in seguito ad una selezione dell'utente, dobbiamo mostrare i nuovi risultati. Per fare questo eliminiamo dal result-set tutte quelle tuple che contengono istanze che non corrispondono a quella dell'oggetto selezionato dall'utente. Se volessimo richiamare il precedente esempio, una volta che l'utente seleziona un particolare museo (per esempio il Museo Nazionale) dobbiamo eliminare

dal result-set tutte quelle tuple che contengono dati riguardanti un museo diverso da quello selezionato dall'utente.
A questo punto dobbiamo trovare la nuova migliore visualizzazione. Per fare ciò rieseguiamo gli algoritmi dell'analisi dinamica e del calcolo dei rank sul nuovo result-set in quanto, siccome il result-set è stato ridotto, le sue caratteristiche statistiche cambiano sicuramente. Tutto questo porterà al calcolo della visualizzazione migliore del nuovo result-set riusando gli algoritmi esistenti. Verosimilmente la migliore visualizzazione non sarà più la stessa di prima e mostrerà attributi diversi dai precedenti.
Le visualizzazioni innestate sono quindi una serie di visualizzazioni che passo dopo passo esplorano i dati andando sempre più in profondità, selezionando non più solo dati generici ma relativi a particolari istanze che vengono selezionate dall'utente.
In precedenza abbiamo detto che se l'utente non effettua nessuna selezione sui dati le visualizzazioni devono essere omogenee. Le visualizzazioni innestate sono però pensate appositamente per permettere all'utente di operare delle selezioni sui dati.
In questo caso ha quindi senso mostrare, in una visualizzazione di attributi di un certo oggetto anche altri attributi appartenenti ad altri oggetti, a patto che, tali altri oggetti siano stati selezionati dall'utente per la costruzione della visualizzazione considerata.

2.1.5 Mapping sui tipi di widget

Fino ad ora abbiamo descritto come generiamo una visualizzazione soffermandoci in particolare su quali attributi degli oggetti visualizzare in ognuna di esse. Nel capitolo 1 abbiamo detto che il SeCo è già provvisto di un'interfaccia grafica per la visualizzazione dei dati. Questa interfaccia contiene alcuni widget di vari tipi come mappe, grafici cartesiani, liste, timeline ecc... Tutti questi oggetti sono dei moduli indipendenti tra loro e altamente personalizzabili.
L'ultimo tassello che manca tra gli algoritmi che abbiamo presentato fino ad ora e l'interfaccia grafica del SeCo è un algoritmo che, date le visualizzazioni

generate decida quale tipo di widget deve essere utilizzato per presentare i dati all'utente finale. In questo paragrafo descriveremo la logica di quest'ultima fase.

Questo algoritmo deve essere eseguito per ogni singola visualizzazione generata per cui si vuole determinare il tipo di widget da utilizzare per mostrare i dati. Questo procedimento prende come input una visualizzazione generata e si occupa di eseguire un'analisi semantica sui tipi degli attributi che la compongono.
In base ai tipi semantici contenuti nella visualizzazione considerata viene scelto un particolare tipo di widget. Per esempio se una visualizzazione mostra sugli assi X e Y la latitudine e la longitudine di un certo oggetto verrà scelto il tipo mappa.
Questo algoritmo si occupa non solo della scelta del tipo di widget ma anche di mappare ogni singola dimensione della visualizzazione sulle dimensioni effettive del tipo di widget scelto. Potrebbe anche accadere che un certo tipo di widget non abbia tutte le 7 dimensioni definite nel nostro modello di visualizzazione (un eventuale grafico a barre avrà di certo a disposizione una dimensione sulla quale mappare il nostro "Clue shape").
Abbiamo definito 6 tipi generici di widget:

Mappa Rappresenta il generico widget attraverso cui i dati vengono visualizzati su di una cartina geografica. Può visualizzare tutte le 7 dimensioni da noi definite tranne l'asse temporale e viene utilizzata quando la visualizzazione che deve rappresentare contiene sui primi due assi attributi di geolocalizzazione;

Grafico a dispersione Rappresenta il generico widget in cui i dati vengono visualizzati in un spazio cartesiano. Anche in questo caso possono essere rappresentate tutte le nostre 7 dimensioni tranne l'asse temporale. Questo widget viene utilizzato quando abbiamo attributi numerici sulle prime due dimensioni della visualizzazione.

Grafico linea Rappresenta un caso particolare del grafico a dispersione in cui i dati dell'asse delle ascisse appartengono ai tipi semantici statistica, distanza, campo calcolato o prezzo. Abbiamo scelto questi tipi semantici in quanto essi, molto probabilmente, assumeranno valori continui. Con questo widget è possibile mostrare "l'andamento" di un certo attributo in funzione di un altro appartenente ad un dominio continuo.

Grafico a barre Rappresenta il generico widget in cui vengono visualizzate delle barre verticali la cui lunghezza è proporzionale al valore del dato da rappresentare. Sull'asse X verranno rappresentati attributi che sono per loro natura discreti (ad esempio nome o tipo di ristorante) oppure attributi i cui valori risultano essere altamente clusterizzabili. Su questo grafico non possono essere rappresentate le dimensioni asse temporale, forma, grandezza e colore del marker (clue shape e clue size) definite nel nostro modello di visualizzazione. Infine non ha senso utilizzare il campo info se sull'asse X sono presenti degli attributi che non sono identificativi.

Elenco Questo è un tipo di widget che banalmente rappresenta un elenco di valori di qualsiasi tipo. In questo caso avremmo a disposizione una sola dimensione su cui mappare i nostri assi X o Y. Esistono poi le dimensioni colore, forma e dimensione.

Linea temporale È un tipo di widget adatto a rappresentare dati caratterizzati da un certo ordine cronologico.

L'output di questo algoritmo sarà quindi il tipo di widget da utilizzare per mostrare all'utente attraverso un'interfaccia grafica una certa visualizzazione più una serie di parametri di configurazione quali per esempio la scala e il range di un certo asse.
Nel caso in cui la visualizzazione considerata contenga uno o più attributi assegnati a dimensioni non disponibili nel widget selezionato bisogna fare una scelta: o non si mostra l'attributo in questione, oppure lo si aggiunge a quello già assegnato alla dimensione info. La dimensione info è infatti sempre

presente in qualsiasi widget e può essere considerata come suddivisa in "slot" all'interno dei quali possono essere mostrati degli attributi.

Aggregazione di visualizzazioni

Durante la progettazione degli algoritmi è stata presa in considerazione l'ipotesi di aggregare più visualizzazioni.
Per aggregazione di visualizzazioni intendiamo che una o piu visualizzazioni dello stesso tipo (mappa, grafico a barre, ecc..) già generate potrebbero essere visualizzate insieme in un unico widget.
La fase di aggregazione può essere applicata in due ambiti:

- Nel caso in cui siano state generate diverse visualizzazioni poco informative. In questa situazione è utile aggregare le diverse viste per aumentare la quantità di informazione che viene trasmessa all'utente. In questo caso l'aggregazione viene effettuata solo su viste che mostrano attributi appartenenti allo stesso oggetto;

- Il secondo (figura 2.4) sarebbe quello di aggregare visualizzazioni anche relative ad oggetti diversi dello stesso result-set in modo da permettere all'utente di poter osservare a colpo d'occhio eventuali relazioni tra le diverse entità.
  Facendo questo tipo di aggregazione l'utente potrà avere una visione leggermente più complessa, ma molto più sintetica sull'intero result-set in un'unica visualizzazione.

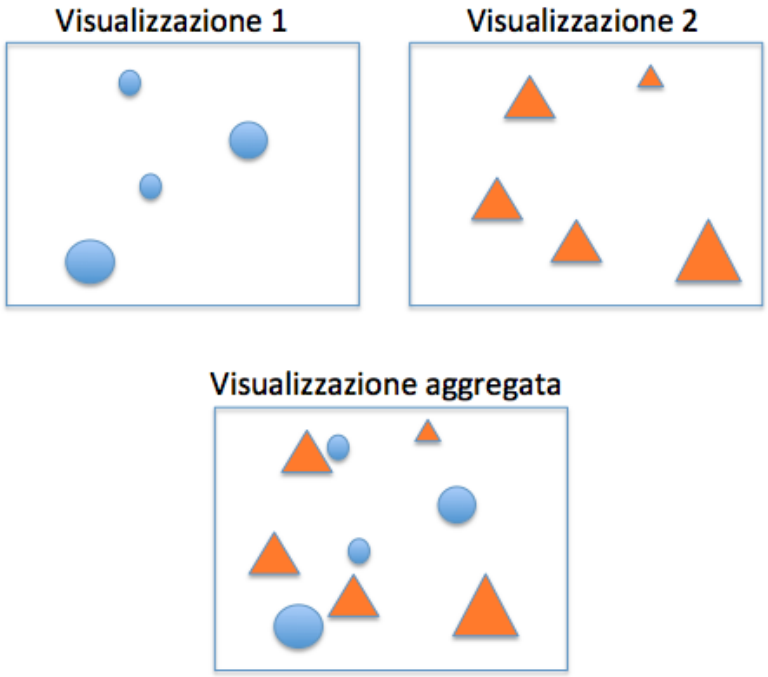

Figura 2.4: Visualizzazioni aggregate

Per quanto riguarda questa ipotesi, come illustreremo nel paragrafo 3.2.3, attraverso il test svolto coinvolgendo diverse persone è emerso che l'aggregazione è un'opzione che sarebbe accolta molto favorevolmente dagli utenti.

Considerato ciò, possiamo dire che in futuro, la soluzione proposta dovrà evolversi in modo da considerare la possiblità di aggregare tra loro le visualizzazioni generate.
In altre parole gli algoritmi che abbiamo sviluppato andranno mantenuti e, prima della configurazione degli widget, bisognerà analizzare le visualizzazioni che vengono generate ed eventualmente configurare il mapping degli widget in modo che l'interfaccia utente mostri più visualizzazioni aggregate.
Nel caso si aggreghi con lo scopo di accresecere il potere informativo delle viste, bisognerà semplicemente unire le visualizzazioni caraterizzate da un rank basso. Questa operazione è già stata definita nella nostra soluzione per le visualizzazioni di tipo "Elenco".
Nel caso in cui si aggreghi con lo scopo di sintetizzare i dati riguardanti di-

verse entità, sarà necessario rinunciare ad una delle dimensioni del marker (forma o colore) delle singole visualizzazioni per differenziare le istanze appartenenti alla diverse entità.
A questo punto però, bisogna considerare la vista aggregata come un unica visualizzazione.

2.2 Generazione di visualizzazioni consecutive

Il secondo obiettivo di questa tesi è quello di creare un algoritmo per proporre all'utente, dato un certo result-set ottenuto da una ricerca effettuata, una serie di visualizzazioni consecutive che gli permetta di avere sui dati più punti di vista che siano però connessi tra loro.
La visualizzazione migliore che viene proposta all'utente utilizzando gli algoritmi presentati fino ad ora mostra, in un ordine ben preciso, solo pochi attributi tra tutti quelli del result-set ottenuto dal motore di ricerca, pertanto ci aspettiamo che sicuramente l'utente sarà interessato ad osservare i dati sotto una diversa prospettiva o sostituendo qualche attributo a quelli proposti in prima istanza dal nostro algoritmo.
L'obiettivo che ci poniamo quindi in questa seconda parte del nostro lavoro è quello di proporre all'utente una sequenza di visualizzazioni che cerchi di soddisfare questi suoi bisogni. In altre parole vogliamo proporre una serie di visualizzazioni connesse tra loro da un filo logico e che mostrino i dati da più punti di vista. Vogliamo in qualche modo proporre all'utente il "percorso" logico attraverso le possibili visualizzazioni che più si avvicina a come egli vorrebbe esplorare i dati se lo lasciassimo libero di crearsi di volta il volta manualmente ogni singola visualizzazione.
In questo paragrafo viene presentato l'algoritmo che abbiamo creato per generare questi "percorsi di visualizzazioni".

L'algoritmo parte creando per prima cosa un insieme di possibili visualizzazioni per i dati appartenenti al result-set considerato. Questo insieme viene calcolato con lo stesso metodo utilizzato per le visualizzazioni alternative. Esso, non conterrà tutte le possibili visualizzazioni creabili utilizzando result-set corrente, ma solo quelle migliori facendo una valutazione sul valore del rank assoluto associato ad ognuna di esse. Il concetto di rank assoluto verrà introdotto tra poche righe. Il numero di visualizzazioni create viene limitato per due principali ragioni:

- la prima è di natura concettuale: se calcolassimo tutte le possibili visua-

lizzazioni, molte di esse sarebbero insignificanti; con il metodo adottato calcoliamo solo le più significative: l'algoritmo che utilizziamo inizia infatti generando le visualizzazioni più significative e via via genererà visualizzazioni che sicuramente saranno sempre più insignificanti. Questo accade perché per la costruzione delle prime visualizzazioni usiamo gli attributi che occupano le prime posizioni nelle classifiche dei rank, man mano che tale algoritmo viene rieseguito più e più volte verranno utilizzati attributi aventi rank sempre più bassi e di conseguenza le visualizzazioni da essi generate avranno una qualità sempre più scarsa;

- la seconda è di natura computazionale: generare tutte le visualizzazioni possibili combinando tra loro tutti gli attributi del result-set sarebbe un'operazione molto onerosa dal punto di vista computazionale e quindi inattuabile se si pensa che dovrebbe essere eseguita per ogni singola ricerca effettuata dagli utenti.

2.2.1 Costruzione del grafo di navigazione delle visualizzazioni

L'obiettivo è quello di costruire un grafo avente come nodi le visualizzazioni contenute nell'insieme generato. Gli archi collegano tra loro le possibili visualizzazioni definendo il concetto di successione tra due viste. Il "percorso" di visualizzazioni da proporre all'utente verrà creato proprio su tale grafo.

Una volta creato l'insieme delle visualizzazioni generabili dal result-set corrente, dobbiamo trovare la visualizzazione migliore dalla quale partirà il "percorso" che proporremo all'utente. Per fare questo abbiamo adottato un'euristica che ci permette di assegnare un rank ad ogni singola visualizzazione. Questo punteggio di rank sarà chiamato "rank assoluto" in quanto esso viene calcolato individualmente per ogni visualizzazione. Nel calcolo di questo parametro vengono utilizzate informazioni appartenenti esclusivamente alla visualizzazione stessa. In altre parole le caratteristiche di una visualizzazione non influenzano i rank assoluti delle altre.
L'euristica che abbiamo adottato per calcolare questo rank è molto sempli-

ce, è la media dei rank di tutti gli attributi assegnati alle dimensioni della visualizzazione considerata.

$$R_a = \frac{\sum_{i=1}^{N} rank_i}{N} \tag{2.9}$$

Dove N è il numero di dimensioni della visualizzazione considerata mentre $rank_i$ è il rank dell'attributo associato alla dimensione *i-esima* della visualizzazione. Di primo acchito tale euristica sembrerebbe troppo semplicistica rispetto alla responsabilità che le viene affidata: deve esprimere una valutazione complessiva di un'intera visualizzazione; ricordiamo che per valutare un singolo attributo nella prima parte di questa tesi utilizzavamo un metodo molto più complesso. In realtà però, il potere informativo di un'intera visualizzazione è strettamente correlato con quello dei singoli attributi associati alle sue dimensioni.
Una volta calcolato il valore del rank assoluto per ognuna delle visualizzazioni appartenenti all'insieme generato possiamo determinare la migliore semplicemente selezionando quella che ha ottenuto il massimo rank assoluto. Questa visualizzazione è il punto di partenza del percorso che proponiamo all'utente. Ora abbiamo il primo passo del percorso. In questo primo passo definiamo questa visualizzazione come "visualizzazione corrente". Nella figura (2.5) abbiamo le visualizzazioni appartenenti all'insieme calcolato. Esse sono rappresentate dai cerchietti azzurri, al loro interno viene riportato il rispettivo rank assoluto. Tra le visualizzazioni abbiamo ora selezionato la migliore e la definiamo "visualizzazione corrente".
A questo punto, tra le visualizzazioni appartenenti all'insieme generato all'inizio, dobbiamo selezionare quali di esse sono candidate ad essere successive a quella corrente. Definiamo che un arco del grafo rappresenta una successione tra due visualizzazioni. La visualizzazione a cui punta un arco è una candidata sucessiva di quella corrente. Ora, sul grafo, dobbiamo quindi creare gli archi percorribili tra la visualizzazione corrente e le candidate successive.
In questo modo, si può capire come, il grafo venga generato passo dopo passo proprio mentre viene esplorato. Per specificare meglio quest'ultima afferma-

zione possiamo dire che i nodi del grafo esistono fin da subito, dal momento in cui vengono generate le visualizzazioni dell'insieme iniziale, mentre gli archi vengono creati passo dopo passo mano a mano che il grafo viene esplorato.

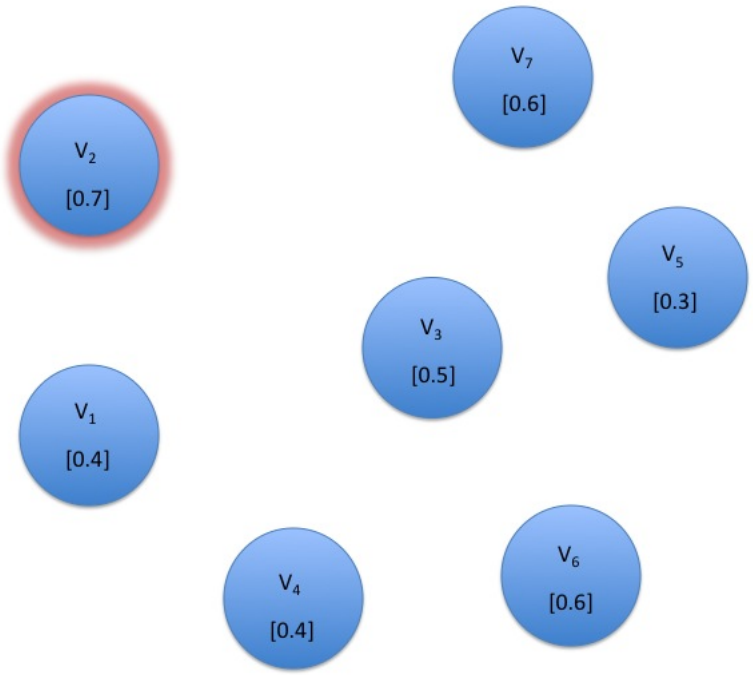

Figura 2.5: Insieme delle visualizzazioni

Abbiamo definito che una visualizzazione V_B è successiva ad una seconda visualizzazione V_A se sussiste almeno una delle seguenti condizioni:

- V_B mostra un attributo che viene mostrato anche da V_A;
- V_B mostra un attributo appartenente allo stesso oggetto a cui appartiene un attributo mostrato nella visualizzazione V_A;
- V_B mostra un campo calcolato per aggregazione riferito ad uno o più attributi mostrati nella visualizzazione V_A.

Durante la costruzione-esplorazione del grafo, nei passi successivi al primo, si aggiunge una ulteriore ed indispensabile condizione: una visualizzazione V_B per essere una candidata successiva di una visualizzazione V_A non deve

essere mai stata utilizzata nei passi precedenti a quello corrente. Quest'ultima condizione viene imposta per due motivi:

- evitare di proporre all'utente visualizzazioni che ha già esplorato;
- evitare di creare dei cicli all'interno del grafo.

Ora nel nostro grafo, mostrato nella figura 2.6 abbiamo quindi creato una serie di archi che partono dalla visualizzazione corrente e arrivano a tutte le visualizzazioni che sono candidate ad essere successive ad essa.

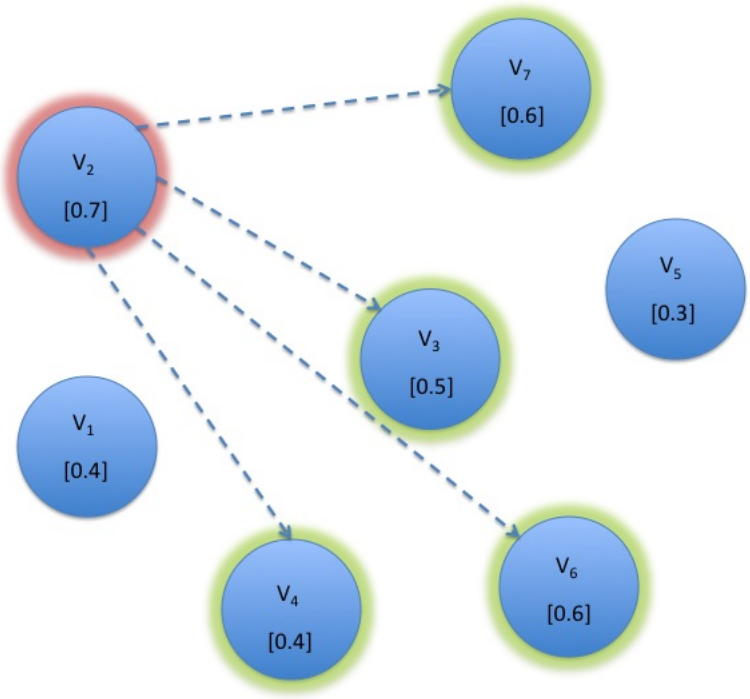

Figura 2.6: Selezione delle visualizzazioni candidate successive

2.2.2 Costruzione e valutazione degli archi del grafo

A questo punto introduciamo un nuovo parametro che chiameremo rank relativo. Mentre il rank assoluto delle visualizzazioni viene utilizzato per valutare una visualizzazione indipendentemente da tutto ciò che nel grafo le sta attorno, tenendo conto soltanto delle sue caratteristiche, questo nuovo rank valuta la visualizzazione in funzione anche delle visualizzazioni che lungo il

percorso che stiamo costruendo sono già state selezionate. Questo rank è quell'elemento che ci permette di valutare quantitativamente quanto sia utile e quanto abbia senso mostrare, al passo corrente, e dopo aver già mostrato un certo insieme di attributi la visualizzazione considerata.
Questo "rank relativo" viene utilizzato nel grafo come peso associato ad un arco che collega tra loro due visualizzazioni.
L'euristica che utilizziamo per calcolare questo rank è la seguente:

$$R_r = \sum_{i=1}^{M}[(n-1-k_i)*(rank_i)]*(1-w)+(rank_s)*w \tag{2.10}$$

Diamo ora una definizione dei termini utilizzati nella formula (2.10). Questa formula viene calcolata per ognuna delle visualizzazioni candidate ad essere successive di quella corrente.
La sommatoria va da 1 a M dove M è il numero di attributi mostrati dalla visualizzazione considerata.
n è il passo a cui si trova l'algoritmo rispetto al percorso che sta generando. Stiamo calcolando i rank relativi per determinare la visualizzazione successiva alla *n-esima*. La visualizzazione corrente è l'*n-esima*.
k_i è il numero di volte che l'attributo preso in considerazione è già stato utilizzato lungo il percorso nei passi precedenti.
$Rank_i$ è il rank dell'*i-esimo* attributo considerato di volta in volta ad ogni passo della sommatoria. Questo rank è già stato calcolato dall'algoritmo dedicato all'individuazione della migliore visualizzazione.
$Rank_s$ è il valore massimo tra tutti i rank degli attributi (definiti attributi di successione) che hanno permesso di individuare la visualizzazione considerata come successiva di quella corrente. In altre parole è il massimo rank tra quelli degli attributi che hanno in comune la visualizzazione corrente e la candidata successiva considerata.
Infine w è un peso che può assumere valori che vanno da 0 a 1 e serve per bilanciare le due parti dell'euristica:

- la prima parte dell'euristica è dedicata a valutare la qualità della visualizzazione successiva considerata rispetto a quelle già mostrate. Ad

ogni passo successivo al primo vogliamo che ogni singola visualizzazione mostri informazioni nuove rispetto a ciò che l'utente ha già avuto modo di osservare nei passi precedenti del percorso. Per questo con la sommatoria penalizziamo le visualizzazioni che mostrano attributi già utilizzati in precedenza;

- la seconda parte dell'euristica invece è stata pensata per valutare la qualità della successione: essa vuole valutare quanto abbia senso approfondire la visualizzazione corrente rispetto ad un attributo piuttosto che un altro.

Nell'euristica, deve avere più forza la seconda parte in quanto l'obiettivo è quello di proporre all'utente un percorso di visualizzazioni connesse tra loro. A questo punto siamo quindi in grado di calcolare per ogni possibile successione il rank relativo. Il rank relativo verrà utilizzato nel grafo come peso di ogni singolo arco.

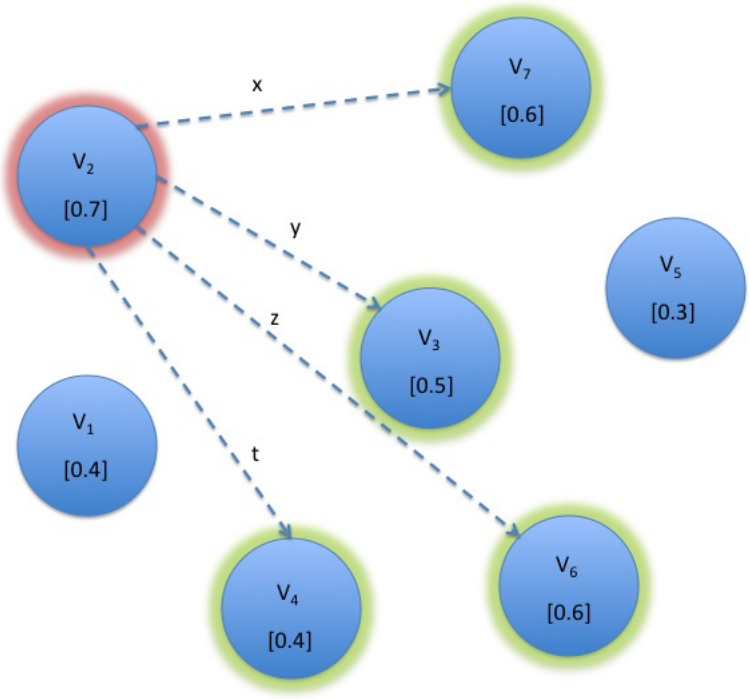

Figura 2.7: I rank relativi vengono utilizzati come pesi (in questo caso x, y, z e t) posti sui rispettivi archi

2.2.3 Selezione di un nodo

Nella figura 2.7 le lettere x,y,z e t rappresentano i rank relativi calcolati per ogni arco tra la visualizzazione corrente e le candidate successive.
Fatto ciò dobbiamo decidere, con un metodo quantitativo, quale visualizzazione scegliere come successiva tra le candidate. In altre parole dobbiamo, nel grafo, selezionare il miglior nodo successivo.
Per fare questo abbiamo adottato un metodo greedy principalmente per due fondamentali motivi:

- abbiamo detto che costruiamo il grafo, o meglio creiamo gli archi tra i nodi, passo dopo passo durante l'esplorazione. Quindi abbiamo a disposizione gli archi per poter scegliere solo il nodo immediatamente successivo a quello corrente; non abbiamo modo di poter calcolare l'intero percorso a priori in quanto non esistono ancora tutti gli archi che connettono tra loro i vari nodi, le varie visualizzazioni;

- la seconda ragione è che a noi non interessa trovare tutto il percorso a priori in quanto ad ogni passo vogliamo esplorare quello che è, al momento, l'insieme di possibili visualizzazioni successive e tra quelle selezionare la migliore.

Quindi ora, dobbiamo scegliere la miglior visualizzazione successiva sulla base dei rank assoluti e relativi che abbiamo a disposizione, per fare questo calcoliamo, per ogni possibile successore un punteggio p:

$$p = v * (R_r) + (1 - v) * (R_a) \tag{2.11}$$

Dove R_r è il rank relativo calcolato con la formula (2.10) e R_a è il rank relativo calcolato con la formula (2.9). v è un parametro che serve a bilanciare le due parti dell'euristica, in particolare vogliamo dare più importanza al parametro che valuta la successione. Abbiamo quindi pensato di assegnare a v il valore 0.6.
Il punteggio p calcolato non è altro che una media pesata tra il rank relativo della successione tra le due visualizzazioni e del rank assoluto della visualizzazione candidata successiva considerata.

A questo punto scegliamo la visualizzazione successiva che ha ottenuto il massimo punteggio p (figura 2.8).

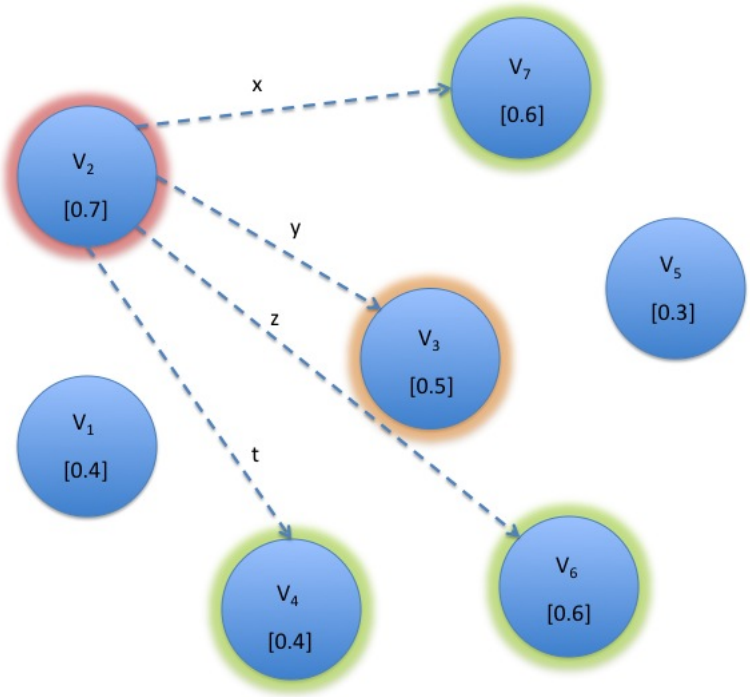

Figura 2.8: Selezione della visualizzazione successiva

Con il supporto della figura 2.9 possiamo quindi notare che ora nel grafo è possibile selezionare la visualizzazione successiva migliore. Quest'ultima diventa quindi la "nuova" visualizzazione corrente.
E' terminato un passo, l'algoritmo può ora ripartire dalla visualizzazione selezionata per trovare le nuove candidate visualizzazioni successive escludendo quelle già selezionate nel percorso fino ad ora compiuto.

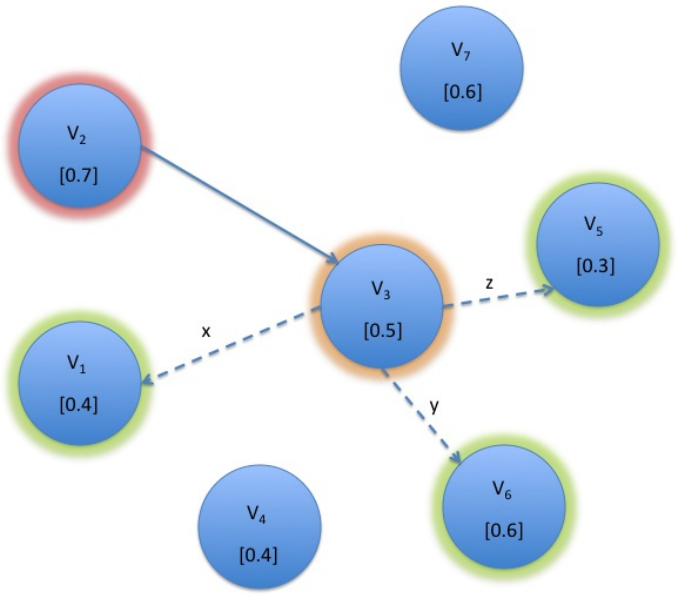

Figura 2.9: L'algoritmo riparte per trovare le nuove visualizzazioni successive

Alla fine l'algoritmo produrrà un'intero percorso tra le visualizzazioni che verrà proposto all'utente finale (figura 2.10).

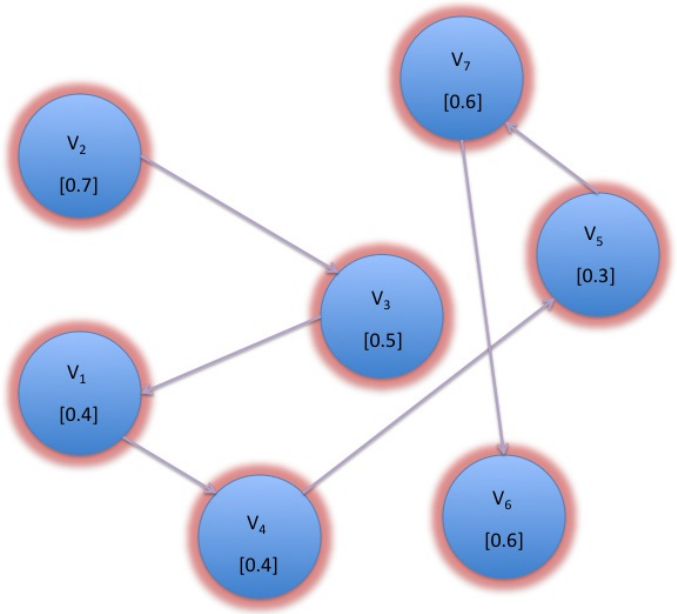

Figura 2.10: Percorso completo

L'utente può ora navigare attraverso il percorso di visualizzazioni proposto. Egli ha così la possibilità di esplorare i dati da più punti di vista che vengono generati e proposti automaticamente nell'ordine più opportuno. Durante l'esecuzione e la generazione di tutto questo scenario all'utente non viene richiesta nessuna attività, non deve configurare nessun aspetto dell'interfaccia, è tutto automatico.

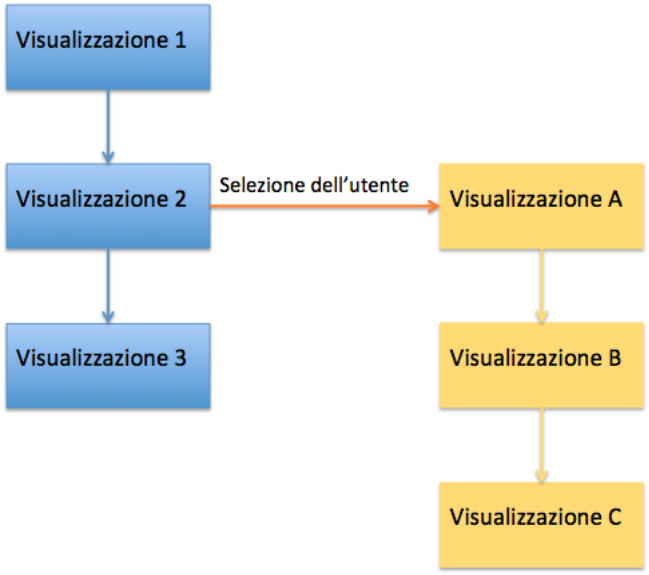

Figura 2.11: Percorso con interazione dell'utente

La navigazione tra le visualizzazioni non è però limitata alla sola consultazione dei dati: l'utente se vuole può interagire con il percorso generato. Può infatti decidere, arrivato ad un certo passo del percorso, di selezionare una particolare istanza di un oggetto, in questo caso le visualizzazioni successive non verranno più mostrate: viene invece calcolata e mostrata una visualizzazione innestata che approfondisce i dati riguardanti l'oggetto selezionato dall'utente. La nuova visualizzazione innestata viene generata dopo aver ristretto result-set alle sole tuple che contengono dati riguardanti quell'oggetto. A questo punto è anche possibile far ripartire l'intero algoritmo della generazione del percorso forzando, come prima visualizzazione, la visualizzazione innestata che è stata generata in seguito alla selezione dell'utente (figura 2.11).

Capitolo 3

Implementazione ed esperimenti

Questo capitolo si divide principalmente i due parti: implementazione e validazione.
Nella prima parte descriviamo come concretamente abbiamo implementato gli algoritmi definiti nel capitolo precedente giustificando tutte le scelte più importanti.
Nella seconda parte invece presentiamo i metodi che abbiamo adottato per testare e validare gli algoritmi implementati.

Abbiamo implementato gli algoritmi creando un prototipo stand-alone fuori dal reale workflow del SeCo: in pratica simuliamo che ci vengano forniti dal motore di ricerca un result-set e il relativo result-model, il nostro prototipo esegue gli algoritmi e genera un output in cui vengono descritte le visualizzazioni alternative, le innestate ed il percorso di visualizzazioni da proporre all'utente.
Per quanto riguarda le tecnologie utilizzate per realizzare questo prototipo abbiamo usato il linguaggio Java per l'implementazione degli algoritmi in quanto è il linguaggio con cui è stata sviluppata la parte server-side del SeCo. Così facendo il nostro prototipo potrà essere integrato nel SeCo facilmente e velocemente.
Abbiamo inoltre utilizzato un database MySQL per simulare lo scambio di dati con il SeCo, leggiamo da database i dati che il nostro prototipo si aspet-

ta in input e scriviamo ciò che dobbiamo fornire in output al resto del sistema.

3.1 Implementazione

Descriviamo ora l'architettura della nostra soluzione e gli aspetti più significativi dell'implementazione per ognuna delle "sezioni" principali degli algoritmi illustrati nel capitolo 2.

3.1.1 Architettura

Il modulo che abbiamo progettato dovrà essere visto dal SeCo come una scatola nera che da una parte si deve interfacciare con ciò che già esiste della parte server-side del SeCo ricevendo in input i risultati della ricerca eseguita, dall'altra deve inviare all'interfaccia grafica del client alcuni parametri che definiscono le modalità di visualizzazione dei dati.
Il prototipo che abbiamo realizzato per implementare e testare gli algoritmi che abbiamo ideato simula la ricezione dei result-set e result-model leggendo i dati da un database e genera le visualizzazioni inserendole poi in un'altra tabella nello stesso database. Abbiamo sviluppato un tool Web, che per ogni tipo di widget simula, per scopi dimostrativi, l'interfaccia grafica del SeCo. L'intero prototipo si divide in cinque parti principali:

- Analisi statica
- Analisi dinamica
- Ranking
- Mapping visualizzazioni
- Percorsi di visualizzazione

Tutta questa architettura in cui è organizzato il codice che implementa i nostri algoritmi è coadiuvata da una struttura dati pensata ad hoc e che viene illustrata qui di seguito.

3.1.2 Strutture dati

Il result-set letto da database necessita di essere trasferito in una struttura dati adatta per poter accedere in modo pratico dal codice Java ai dati che essa contiene. Per questo motivo abbiamo creato una struttura avente lo schema rappresentato nella figura 3.1.

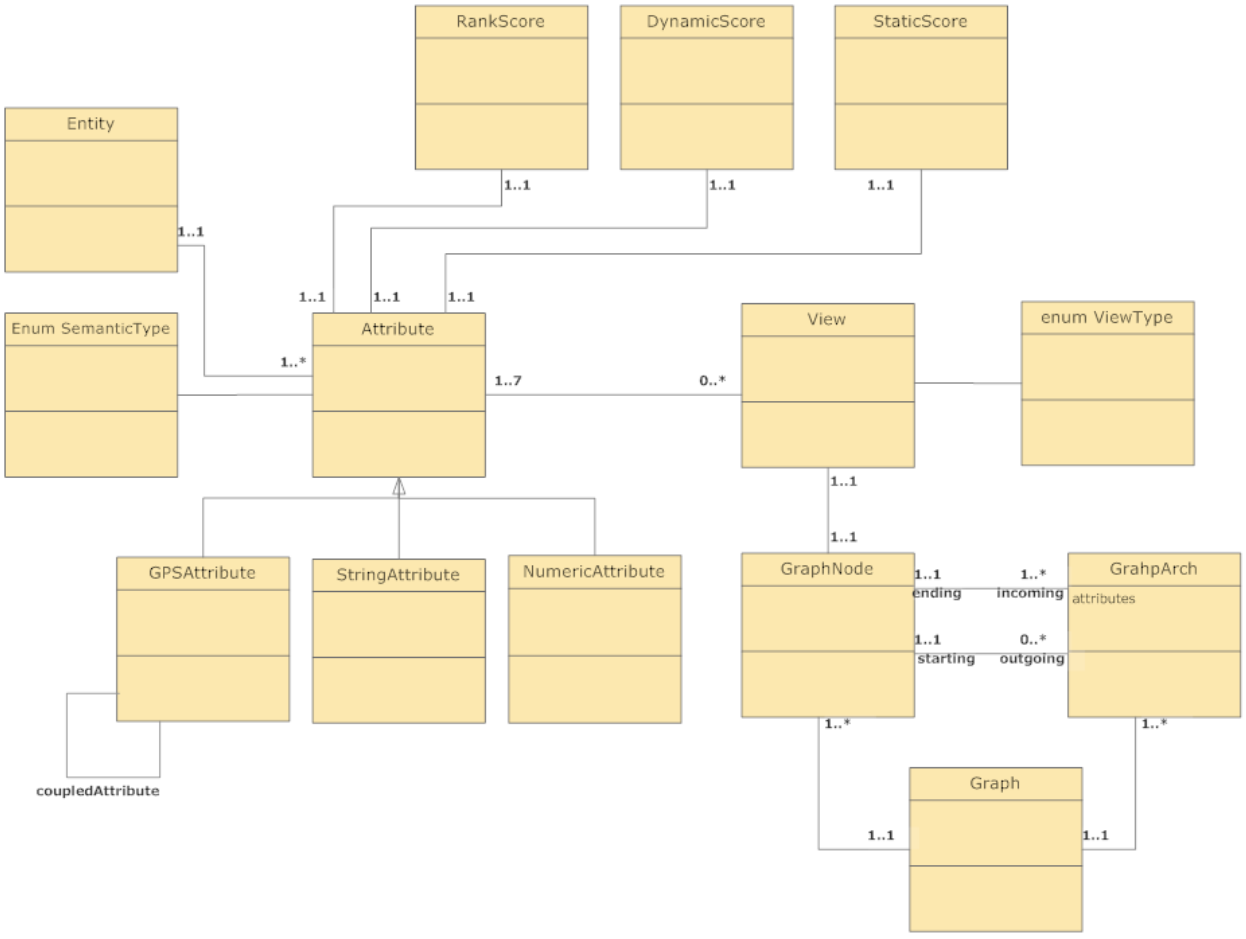

Figura 3.1: Class diagram struttura dati

Le principali entità di questa struttura sono “Attribute”, “View” e “GraphNode”, esse rispettivamente rappresentano un generico attributo appartenente ai vari oggetti contenuti in un result-set, una generica visualizzazione da proporre all’utente, e un nodo generico del grafo su cui viene costruito il

percorso di visualizzazioni.
Di seguito descriviamo nel dettaglio le classi riportate nel diagramma.

Attribute Questa è una classe astratta e rappresenta il generico attributo di un oggetto presente nel result-set. Gli attributi più significativi di questa classe sono:

- **nome**: è un attributo di tipo stringa che rappresenta il nome dell'attributo considerato così come è indicato nel result-set;
- **entità di riferimento**: identifica l'oggetto del result-set a cui appartiene l'attributo considerato;
- **tipo semantico**: questo attributo è di tipo enumeration ed identifica il tipo semantico dell'attributo che l'oggetto rappresenta;
- **staticScore, dynamicScore, rankScore**: questi sono oggetti che contengono i punteggi ottenuti rispettivamente dall'analisi statica, dall'analisi dinamica e dalla fase in cui viene calcolato il rank;

GPSAttribute Questa è una classe che eredita da *Attribute* e rappresenta un particolare attributo che contiene una coordinata GPS. L'unico attributo significativo di questa classe, oltre a quello che contiene il valore della coordinata, è *coupledAttribute*, anch'esso di tipo *GPSAttribute* che indica l'altro attributo insieme al quale l'oggetto considerato forma una posizione geografica (Latitudine, Longitudine).

NumericAttribute Anche questa classe eredita da *Attribute* e rappresenta un generico attributo di tipo numerico.

StringAttribute Anche questa classe eredita da *Attribute* e rappresenta un generico attributo di tipo stringa.

Entity Questa classe rappresenta un oggetto contenuto nel result-set. L'attributo più significativo è quello che indica quale campo, all'interno del result-set, è l'id dell'ggetto considerato.

DynamicScore Gli oggetti istanze di questa classe contengono i valori calcolati durante l'analisi dinamica: *Clustered distribution quality* e *Unclustered distribution quality.*

StaticScore Gli oggetti istanze di questa classe contengono i valori ricavati durante l'analisi statica che sono quelli elencati nella sezione 2.1.1.

RankScore Gli oggetti istanze di questa classe contengono tutti i valori di rank calcolati per ogni dimensione delle visualizzazioni.

View Gli oggetti istanze di questa classe rappresentano le visualizzazioni che vengono generate dagli algoritmi. I tre attributi più significativi sono:

- **rank assoluto**: è un attributo di tipo numerico e indica il rank assoluto calcolato per la visualizzazione considerata;
- **arrayAttribute**: è un array di tipo *Attribute* che contiene tutti gli attributi selezionati per la visualizzazione considerata;
- **tipo**: indica il tipo di widget che è stato scelto per la visualizzazione considerata.

Graph è una classe che rappresenta il grafo sul quale viene costruito il percorso di visualizzazioni che viene proposto all'utente. Il grafo concettualmente è composto da nodi connessi tra loro da archi. Gli attributi più importanti che un oggetto istanza di questa classe contiene sono:

- **arrayNode**: è un array che contiene i riferimenti a tutti i nodi nel grafo. Ogni nodo è di tipo *GraphNode*;
- **arrayArch**: è un array che contiene i riferimenti a tutti gli archi che connettono tra loro i nodi del grafo;
- **beginNode**: è un attributo di tipo *GraphNode* e rappresenta il nodo iniziale del percorso che è stato costruito sul grafo.

GraphNode è una classe che rappresenta un singolo nodo del grafo. In particolare gli attributi più importanti per gli oggetti istanze di questa classe sono:

- **view**: è un oggetto di tipo *View* e identifica la visualizzazione rappresentata dal nodo considerato del grafo;

- **selezionato**: è un oggetto di tipo *booleano* e serve per indicare se la visualizzazione identificata dal nodo considerato è già stata selezionata dall'algoritmo e fa quindi già parte del percorso che si sta costruendo.

GraphArch è una classe le cui istanze rappresentano concettualmente gli archi del grafo che connettono tra loro due nodi. I principali attributi di questa classe sono:

- **beginNode**: è un oggetto di tipo *GraphNode* e identifica il nodo di partenza dell'arco rappresentato dall'oggetto considerato;

- **endNode**: è un oggetto di tipo *GraphNode* e identifica il nodo a cui arriva l'arco rappresentato dall'oggetto considerato;

- **selezionato**: è un oggetto di tipo *booleano* e serve per segnalare se l'arco è stato selezionato durante la costruzione del percorso da proporre all'utente. Se il valore di questo attributo è *true* allora l'arco di cui fa parte è stato selezionato, quindi le visualizzazioni rappresentate dai nodi di inizio e fine fanno parte del percorso da proporre all'utente ed in particolare sono l'una la successiva dell'altra;

- **relativeRank**: rappresenta il rank relativo che è stato calcolato. Il rank relativo infatti non è associato ad una visualizzazione, ma è un attributo proprio della successione (di una visualizzazione considerata come successiva di un'altra), che quindi coinvolge due visualizzazioni. Abbiamo quindi ritenuto che l'arco fosse l'oggetto più opportuno in cui salvare il valore del rank relativo.

3.1.3 Selezione della migliore visualizzazione

In questa sezione descriveremo come sono state implementate le parti principali dell'algoritmo per la selezione della migliore visualizzazione da proporre all'utente.

Analisi statica

Per quanto riguarda l'analisi statica, nel precedente capitolo abbiamo detto che essa dipende da un certo numero di parametri che vengono stabiliti a priori dal particolare "Expert User" che vuole creare un'applicazione.
Per quanto riguarda il nostro prototipo, abbiamo creato due tabelle nel database: la prima contiene per ogni tipo semantico i relativi parametri definiti per l'analisi statica, mentre la seconda contiene le eventuali configurazioni personalizzate per ogni singolo attributo.
Nel prototipo Java che abbiamo sviluppato esiste un metodo che ha la funzione di eseguire l'analisi statica. Tale metodo legge i parametri di cui abbiamo appena parlato attraverso una query SQL dal database di supporto. Questa query estrae i parametri dell'analisi statica dalla tabella contenente le configurazioni personalizzate, se per un dato attributo non esiste nessuna configurazione personalizzata, le informazioni verranno estratte dalla prima tabella relativa ai tipi semantici.

Analisi dinamica

L'analisi dinamica viene eseguita da un particolare metodo sviluppato appositamente. L'implementazione di questo metodo è costituito da cicli che vanno a leggere i vari valori dalla struttura dati e calcolano i parametri definiti nel capitolo precedente applicando meramente le relative formule matematiche.

Ranking attributi

Il codice sviluppato per questa parte di algoritmo si limita ad applicare le formule matematiche già definite in precedenza per calcolare i rank degli attributi per ogni dimensione della visualizzazione.

Mapping dei dati

Questa parte del prototipo ha l'obiettivo, dati i rank calcolati nella sezione precedente, di assegnare i vari attributi del reslut-set alle visualizzazioni da proporre all'utente.
Tutto il codice relativo al mapping dei dati è contenuto in una classe Java divisa in due parti: la prima contiene i metodi necessari per generare le visualizzazioni alternative, la seconda quelli per le visualizzazioni innestate.

Visualizzazione dei risultati

Per fini unicamente dimostrativi abbiamo implementato in PHP una piccola applicazione Web che mostra le visualizzazioni generate dal nostro algoritmo. Tale applicazione si serve alcuni widget sviluppati da Google [1] che forniscono gratuitamente le API per utilizzarli e configurarli.
Abbiamo utilizzato le librerie di Google in quanto forniscono tutti gli widget di cui necessitiamo, fatta eccezione per la timeline.
Per scambiare i dati tra il nostro prototipo Java e questa applicazione Web ci serviamo ancora una volta del database di supporto. Il prototipo Java memorizza in quest'ultimo le visualizzazioni generate e le rispettive configurazioni per il widget selezionato. L'applicazione Web legge questi dati dal database e di conseguenza configura il relativo widget. Le API che utilizziamo sono sviluppate in HTML5 e Javascript e non è necessario nessun plug-in per utilizzarle.
Facciamo ora una rapida carrellata tra gli widget che abbiamo selezionato.

Mappa Questo widget visualizza una carta geografica di qualunque zona del Pianeta in 17 diversi livelli di zoom. Sulla carta geografica possono essere posizionati dei "marker" che fungono da segnaposto per indicare un particolare punto geografico. Il marker viene rappresentato attraverso un'immagine che può essere passata come parametro. A questi marker possono essere associate delle piccole finestre che possono contenere qualsiasi tipo di informazione.

[1] http://code.google.com/intl/it-IT/apis/chart/

Su questo widget possono essere mostrate e configurate agevolmente tutte le dimensioni che avevamo definito per il tipo mappa nel paragrafo 2.1.5. Una piccola limitazione è presente per le dimesioni “clueSize”, “clueShape” e “clueColor”: tali parametri infatti non sono direttamente configurabili, possiamo solo utilizzare immagini di colori e dimensioni diverse.

Grafico a dispersione Questo grafico mostra i dati forniti su uno spazio cartesiano, è possibile impostare per ogni asse il valore minimo e massimo. Può essere inoltre configurato agevolmente il colore e la dimensione di ogni punto mostrato sul grafico. Anche in questo widget è possibile associare ad ogni marker una piccola finestra nella quale visualizzare informazioni aggiuntive. Questo widget mostra le dimensioni che avevamo definito per questo tipo di grafico fatta eccezione per la forma del marker.

Tabella Abbiamo scelto di utilizzare questo widget per mostrare gli elenchi in creando una sola colonna. Tra tutti gli widget che abbiamo esaminato questo ci è sembrato il più appropriato per i nostri scopi dimostrativi. Questo widget offre tutte le dimensioni che ci aspettavamo.

Diagramma a barre Questo widget mostra delle barre rettangolari la cui grandezza è proporzionale al valore dei dati a cui si riferiscono. Tale valore è ottenuto applicando un operatore aggregato sui dati appartenenti alla categoria considerata. Per quanto riguarda l’asse verticale è possibile configurare il valore massimo e minimo mentre ciò non è possibile sull’asse orizzontale poichè è destinato a visualizzare attributi discreti o categorici. E’ possibile visualizzare informazioni aggiuntive sulle barre tramite un piccola finestra pop-up.

Diagramma a linea L’utilizzo e la configurazione di questo grafico è del tutto uguale al diagramma a dispersione.

L’unica rappresentazione mancante è quella delle timeline che però posso-

no essere visualizzate utilizzando un grafico a dispersione ignorando l'asse verticale.

3.1.4 Creazione percorso

In questa sezione illustriamo come è stato implementato l'algoritmo che propone all'utente un "percorso di visualizzazioni" dei dati.

Per la creazione del percorso ci serviamo della struttura dati descritta in precedenza (paragrafo 3.1.2) che viene usata per rappresentare un grafo.
Di seguito riportiamo lo pseudocodice dell'algoritmo che abbiamo ideato per la generazione del "percorso".

```
1  //generazione delle visualizzazioni con gli algoritmi precedenti
2  views[] = generateViews()
3  graph[] = {}
4  arcs[] ={}
5  currentView = {}
6  successive[] = {}
7  //creazione dei nodi
8  for t = 0 to views.size(){
9                   node = new Node(views[t])
10                  graph.add(node)
11                  //calcolo dell rank relativo
12                  computeAbsoluteRank(node)
13          }
14 //selezione della visualizzazione migliore
15 currentViewNode = selectInitalView(graph)
16 while(true){
17          //selezione delle candidate successive
18          successive = selectSuccessive{currentViewNode}
19          //se non ci sono successive l'algoritmo l'algoritmo termina
20          if(isEmpty(succesive)){
21                  break;
22          }
23          //creazione degli archi e calcolo del rank relativo
24          for t = 0 to successive.size(){
25                          arch = new Arch(currentViewNode,successive[t])
26                          computeRelativeRank(currentViewNode,successive[t])
27                          archs.add(arch)
28                  }
29          nextViewNode = {}
30          maxRank = 0
31          //selezione della successiva migliore
32          for t=0 to successive.size(){
33                          rank = computeTotalRank(successive[t])
34                          if(rank>maxRank){
35                                  maxRank = rank
36                                  nextViewNode = successive[t]
37                          }
38                  }
39          nextViewNode.selected = true
40          nextViewNode.getIncomingArch.selected = true;
41          //la visualizzazione trovata viene impostata come nuova vista corrente
42          currentViewNode = nextViewNode
43 }
```

L'algoritmo inizia generando l'insieme delle possibili visualizzazioni tra le quali il percorso che proporremo all'utente si potrà articolare.
Per ogni visualizzazione presente nell'insieme viene creato un nodo del grafo. Tutti i nodi del grafo sono oggetti di tipo *GraphNode* e sono memorizzati in un array. A questo punto ancora non esiste alcun arco che collega tra loro i nodi. Gli archi sono oggetti di tipo *GraphArch* e anch'essi sono memorizzati in un array.
L'algoritmo procede calcolando il rank assoluto di ogni visualizzazione, prende il massimo valore trovato e marca la visualizzazione corrispondente come la migliore. Questa visualizzazione selezionata sarà il punto di partenza del percorso che stiamo creando e viene marcata anche come *visualizzazione corrente*.
A questo punto l'algoritmo valuta, verificando le condizioni descritte nel paragrafo 2.2, quali sono le visualizzazioni candidate successive e quindi i nodi del grafo candidati successivi.
Per ogni visualizzazione successiva viene calcolato il rank relativo. Tale parametro viene salvato nell'oggetto che rappresenta l'arco tra la visualizzazione corrente e la candidata successiva considerata.
A questo punto viene calcolata la media pesata tra il rank assoluto e relativo per ogni candidata successiva.
La visualizzazione che otterrà la media pesata più alta verrà marcata (verrà messo a *true* un attributo booleano sia nell'oggetto arco che nell'oggetto nodo considerato) come successiva.
L'algoritmo illustrato fino ad ora è iterativo, viene rieseguito fino a che la visualizzazione considerata come corrente non avrà più visualizzazioni successive.

3.2 Validazione

In questo paragrafo vengono illustrate le metodologie utilizzate per validare gli algoritmi sviluppati.

3.2.1 Dataset utlizzati

Per validare gli algoritmi ideati sono stati utilizzati due result-set che simulano i risultati restituiti dal SeCo effettuando due diverse ricerche.
I result-set sono stati generati sottoponendo al SeCo due query per cercare dati nell'ambito di due diversi scenari:

- Nel primo l'utente ha l'obiettivo di programmare un periodo di studio all'estero, in particolare negli Stati Uniti. Egli vuole cercare un'università in cui studiare e un appartamento in cui alloggiare (scenario 1);

- Nel secondo scenario l'utente vuole organizzare una gita nella città di Roma per visitare un museo. Vuole quindi cercare un museo tra quelli presenti in città, vuole cercare un hotel dove pernottare e un ristorante dove pranzare.(scenario 2).

In questi due scenari di utilizzo si ipotizza di sottoporre al sistema le due query attraverso due diverse applicazioni simulate (impostando i valori dell'analisi statica) che devono essere create da un "Expert User".
L'attività di test che abbiamo svolto è divisa principalmente in due fasi, nella prima abbiamo testato noi stessi il sistema (test interno), mentre nella seconda abbiamo coinvolto altri utenti (test esterno).

3.2.2 Test interno

Questa fase di test è stata condotta durante e dopo lo sviluppo del prototipo. Gli scopi principali di questa prima parte del test svolta durante lo sviluppo sono stati:

- verificare la correttezza dell'implementazione degli algoritmi rispetto a ciò che abbiamo progettato approcciando il problema;
- validare gli algoritmi verificando che i procedimenti logici e matematici che abbiamo ideato funzionassero veramente se applicati ad un caso reale;
- calibrare i valori di tutti i pesi e parametri che abbiamo inserito nelle varie euristiche presentate nel capitolo 2.

La calibrazione è stata particolarmente importante in quanto, facendo i primi test su scenari reali, abbiamo introdotto un cambiamento molto rilevante rispetto a ciò che avevamo progettato. Osservando i primi risultati ottenuti dai test abbiamo infatti modificato l'euristica per il calcolo della "Clustered distribution Quality": ci siamo accorti che con la vecchia versione risultavano valori troppo bassi per gli attributi aventi gruppi di valori le cui cardinalità erano poco omogenee rispetto agli attributi aventi molti gruppi omogenei. Con la vecchia versione poteva capitare che un attributo (per esempio la distanza dalla stazione centrale dell'hotel), la cui distribuzione dei valori è suddivisa in due gruppi con cardinalità che non sono omogenee, assumesse un valore di "Clustered distribution Quality" più alto rispetto ad esempio all'attributo che indica il numero di stelle dell'hotel che presenta un numero di gruppi (6, il valore va da 0 a 5 stelle) maggiore ma certamente è più indicato ad essere clusterizzato.

Dopo questa prima fase, abbiamo condotto un test più realistico, abbiamo simulato di utilizzare il prototipo in due casi d'uso reali descritti dai due scenari appena introdotti. In particolare qui riportiamo un caso d'uso in cui viene utilizzato il primo scenario per testare l'algoritmo per la selezione della miglior visualizzazione, mentre per quanto riguarda l'algoritmo che genera il percorso di visualizzazioni riportiamo un caso d'uso nell'ambito del secondo scenario.
Facendo una ricerca multi-dominio nell'ambito del primo scenario, ipotizziamo che l'utente si aspetti di vedere innanzi tutto le università con le loro

caratteristiche principali in quanto Università è l'entità fondamentale di questo scenario. Questo aspetto è definito "dall'Expert User" (che ha creato l'applicazione) attraverso il parametro *objectTypePriority* dell'analisi statica.
In particolare il prototipo mostra, come illustrato in figura 3.2, le università in una mappa localizzandole sul territorio americano. Sulle dimensioni del marker della mappa vengono mostrate le caratteristiche più importanti di ogni singola università come per esempio il fatto che sia privata o pubblica e il nome che rispettivamente vengono assegnate al colore ed al pop-up informativo.

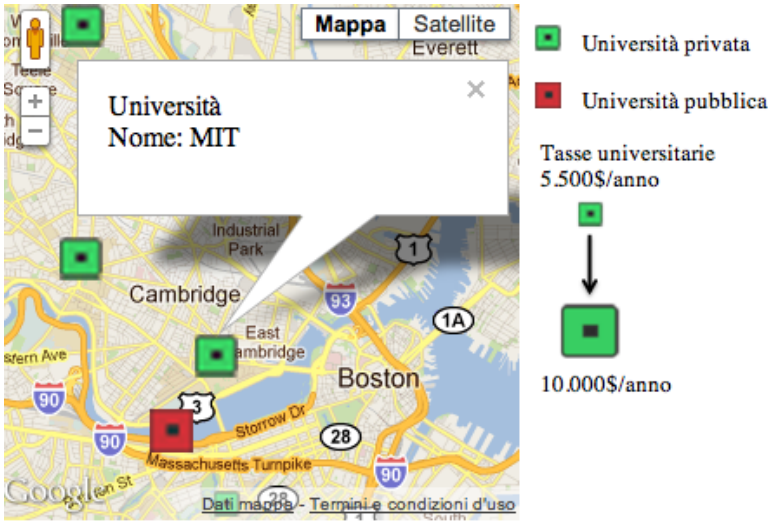

Figura 3.2: Università americane

A questo punto abbiamo ipotizzato che l'utente selezioni una università ed in particolare il MIT. All'utente viene proposta una visualizzazione innestata che mostra gli appartamenti in relazione all'università selezionata. Per ogni appartamento viene mostrata la località geografica in cui si trova (figura 3.3). Il widget scelto è quindi di tipo mappa. E' molto interessante anche citare gli attributi che vengono mostrati sulle dimensioni del marker: al colore viene associato in numero di servizi igienici di cui l'appartamento considerato dispone, alla dimensione viene associato il prezzo. Infine nel pop-up informativo viene visualizzato l'indirizzo postale dell'appartamento.

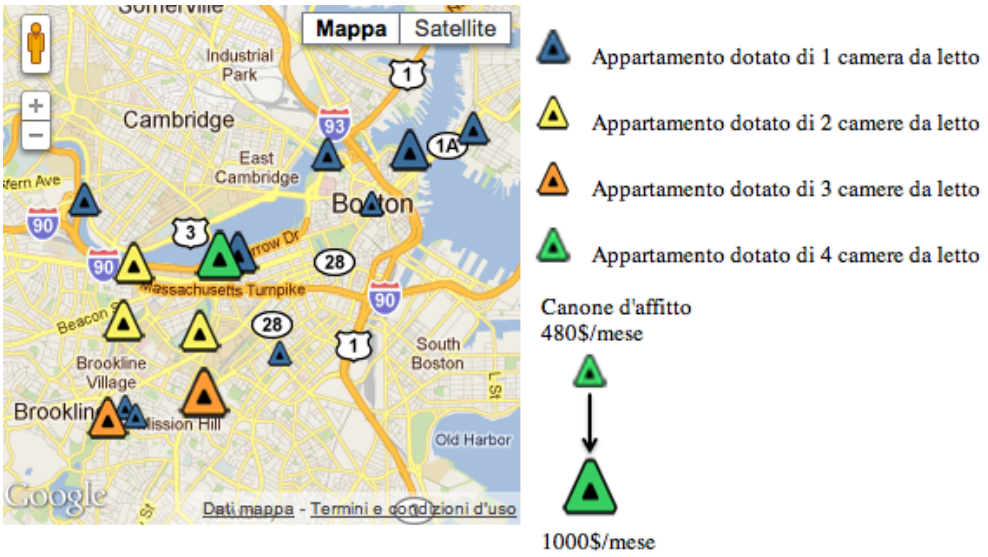

Figura 3.3: Appartamenti

A questo punto, se l'utente seleziona un particolare appartamento vengono mostrate le relative caratteristiche.
Quello illustrato è un caso in cui l'utente esplora le visualizzazioni innestate. Le visualizzazioni innestate permettono all'utente di esplorare i dati soprattutto in profondità, andando a creare delle visualizzazioni sempre più personalizzate riguardo un certo insieme sempre più ridotto e specifico di istanze dei dati.

Un altro modo che viene proposto all'utente per visualizzare i dati è quello delle visualizzazioni alternative.
Viene proposto all'utente un insieme di sei visualizzazioni:

- la prima mostra una mappa con le università ed è uguale alla prima visualizzazione innestata.
- la seconda mostra una mappa con tutti gli appartamenti trovati a prescindere dall'università in quanto l'utente non ha selezionato nessun oggetto;
- poi viene proposta una serie di elenchi aggregati che mostrano il nome dell'università, l'importo delle tasse universitarie, il campo di stu-

dio dell'università, la tipologia (pubblica o privata), la percentuale di ammissioni e la percentuale di borse di studio assegnate (figura 3.4);

Nome	Campi di studio	Percentuale borse di studio	Percentuale ammissione	Spese[$]	Controllo
Boston-College	economics, biology, english	0.6	0.5	10000	private
Boston-University	business-administration, psichology, liberal-arts	0.6	0.6	10000	private
Harvard	history, biology, liberal-arts	0.6	0.2	10000	private
MIT	sciences, electrical-engineering, mechanical-engineering, engineering	0.5	0.3	10000	private
Brandeis	economics, biology, chemistry, pre-med, pre-law, liberal-arts	0.4	0.6	10000	private
Tufts University	liberal-arts, science, engineering	0.45	0.45	10000	private
Smith	education, math, scence, english, social-science	0.3	0.5	8500	private
northeastern	humanities, buiness-administration	0.55	0.8	5500	private

Figura 3.4: Dettagli università

- un grafico a barre (figura 3.5) che rappresenta sull'asse delle x il numero di servizi igienici che possiede un certo appartamento, sulle y il prezzo medio degli appartamenti. Questo è un caso in cui, siccome vengono rappresentati attributi aggregati, non avrebbe senso visualizzare informazioni riguardanti la singola istanza;

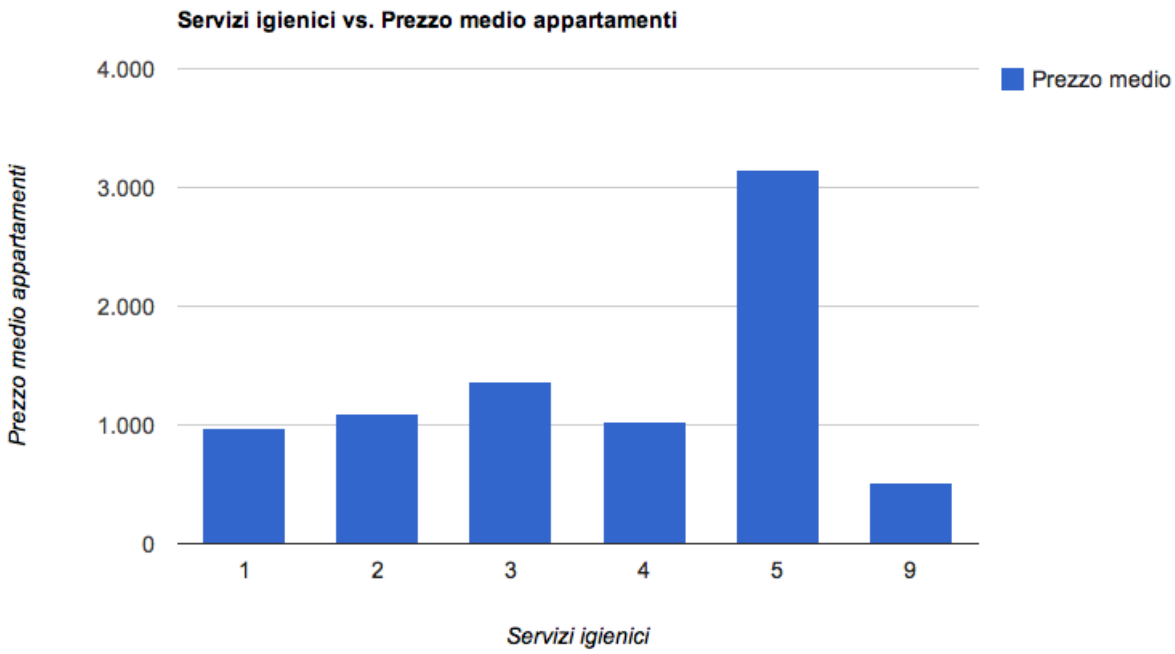

Figura 3.5: Grafico a barre

- un elenco in cui vengono mostrati gli appartamenti e per ognuno di essi il numero di servizi igienici, una foto e l'indirizzo postale;
- l'ultima visualizzazione è anch'essa un elenco che mostra varie informazioni sullo Stato in cui si trovano gli appartamenti e le università trovati come: tasso di omicidi, di rapine, furti d'auto, ecc...

Le visualizzazioni alternative sono tra loro scorrelate in quanto il loro scopo è quello di mostrare varie sfaccettature dei dati da diversi punti di vista.

Descriviamo ora un caso d'uso della generazione automatica di un percorso di visualizzazione sullo scenario riguardante l'organizzazione di una gita nella città di Roma.
Come prima visualizzazione all'utente viene mostrata una mappa (figura 3.6) dei più importanti musei della capitale, di seguito gli viene mostrato un grafico a dispersione che mostra il prezzo del biglietto d'ingresso in funzione dalla

distanza dalla stazione centrale, infine gli viene suggerita la visualizzazione di un elenco che mostra per ogni museo alcuni dettagli, come ad esempio gli orari, se vi è la possibilità di avere una tariffa ridotta, se il museo dispone di un parcheggio privato e se fornisce una rete wireless gratuita. Dopo aver visionato queste informazioni l'utente seleziona un museo, per esempio sceglie di visitare i "Musei Vaticani".

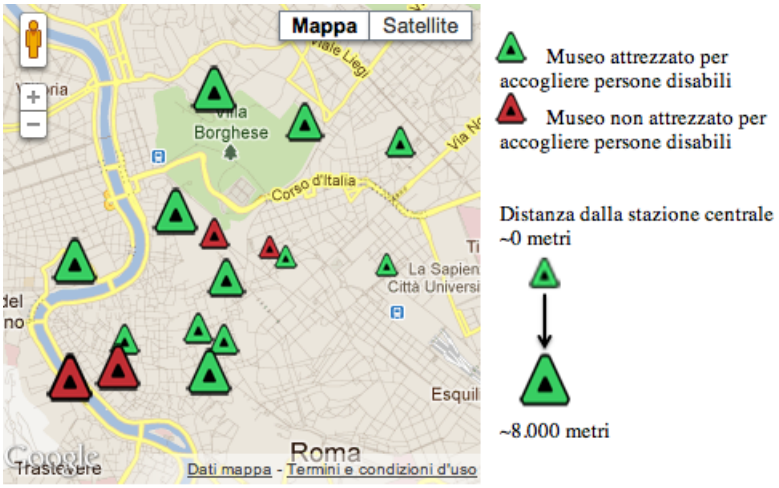

Figura 3.6: Mappa dei musei

All'utente viene quindi suggerito un nuovo percorso di visualizzazione che inizia mostrando la mappa degli hotel nell'intorno del museo selezionato. In questa visualizzazione le dimensioni dei marker riportano i seguenti attributi: la dimensione indica il tempo necessario per raggiungere la stazione centrale, mentre il colore e la forma indicano rispettivamente se l'hotel è fornito di mini-bar in camera e se l'hotel dispone di un parcheggio privato; infine il nome dell'hotel viene associato alla dimensione info.
La visualizzazione successiva proposta nel percorso è di tipo diagramma a dispersione e mostra sugli assi, rispettivamente il prezzo della camera più economica e il tempo necessario per raggiungere la stazione centrale; il colore indica se nell'hotel è presente una sala congressi, la forma indica la presenza della rete wireless e nel campo info viene riportato il nome dell'hotel. La dimensione del marker in questo caso non è associata a nessun attributo.
Nella visualizzazione successiva viene mostrato un diagramma a barre in cui

all'asse x viene associato l'attributo categorico che indica se l'hotel dispone o meno di un parcheggio, mentre alla lunghezza delle barre viene associato il campo calcolato media dei prezzi base degli hotel.
Questa è una visualizzazione che aggrega delle informazioni di più istanze mostrando la media di un prezzo in funzione di una caratteristica dell'hotel. Poi vengono proposti degli elenchi aggregati che mostrano una serie di informazioni come il numero di stelle dell'hotel, la categoria a cui esso appartiene e se è attrezzato per accogliere persone disabili.
A questo punto supponiamo che l'utente abbia selezionato l'hotel "Torino" (figura 3.7).

Nome	Categoria	Punteggio medio	Numero di recensioni	Multilanguage	Presenza di aria condizionata	Si accettano animali
Hotel Delle Province	Hotel	9.2	51	✗	✗	✓
Hotel San Giusto	Hotel	8.4	19	✗	✗	✓
Hotel Regina Margherita	Hotel	8.4	80	✗	✗	✓
Hotel Stromboli	Hotel	7.8	125	✗	✗	✓
Hotel Windrose	Hotel	9	65	✓	✗	✓
Hotel Chicago	Hotel	8.2	33	✓	✓	✗
Hotel Torino	Hotel	8.1	24	✓	✓	✓
Hotel Joli	Hotel	9.1	5	✗	✗	✓
Hotel Marsala	Hotel	7.4	37	✓	✗	✓
Adora Suite B&B	B&B	8.6	20	✓	✗	✓
Domus Livia	Residence	9.2	17	✓	✗	✓
Hotel Center 1-2-3	Hotel	6.5	110	✓	✗	✓
Nautilus	Hotel	8.9	16	✗	✗	✗

Figura 3.7: Tabella che mostra vari dettagli dell'hotel

Come prima visualizzazione viene mostrata una mappa con i ristoranti e le seguenti informazioni attraverso il marker: la dimensione rappresenta il prezzo medio, il colore se è presente o meno la sala fumatori e la forma indica se vengono accettate carte di credito. Nel campo informazioni viene riportato il nome del ristorante.
Come seconda visualizzazione viene proposto un grafico a dispersione che

sugli assi mostra il prezzo medio dei ristoranti in funzione della distanza dalla stazione centrale, il colore indica se il ristorante accetta carte di credito. Infine nel campo info viene indicato il nome del ristorante (figura 3.8).
A questo punto viene proposta una serie di elenchi aggregati che mostrano informazioni quali la categoria del ristorante, il numero di posti disponibili in sala, un campo che indica se è aperto durante la stagione estiva e se è attrezzato per accogliere persone disabili.

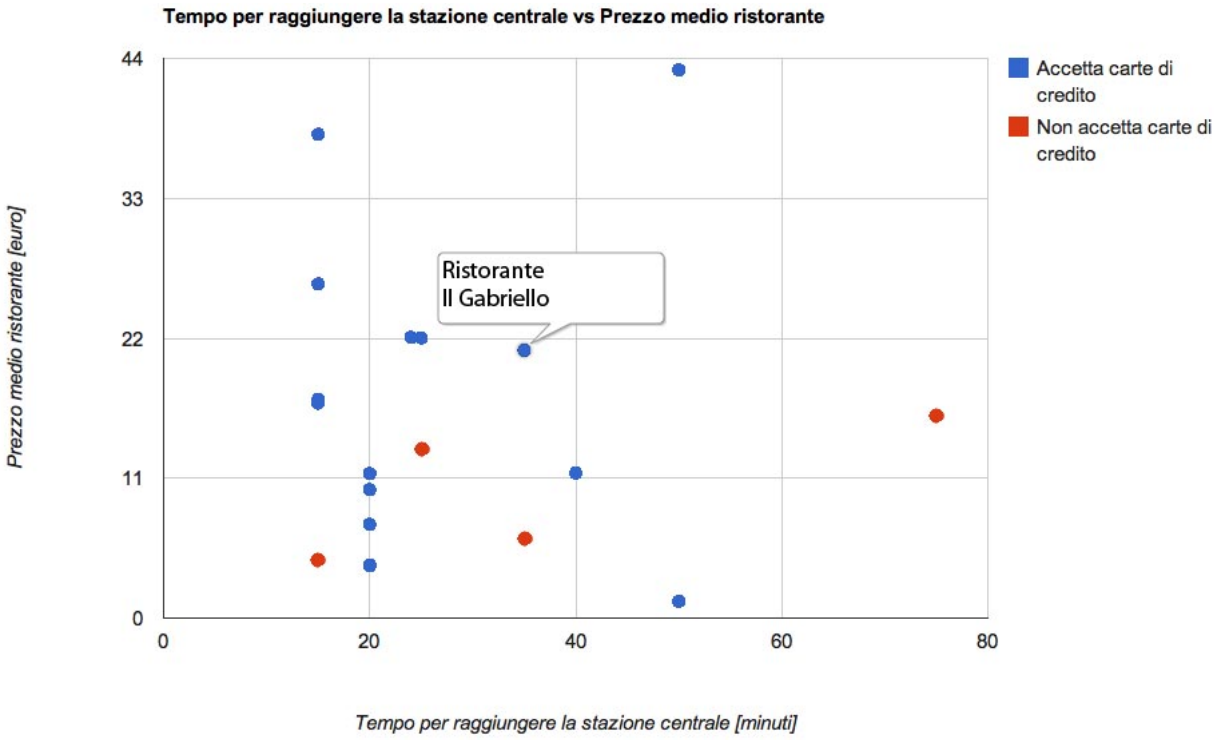

Figura 3.8: Grafco a dispersione che mostra il prezzo medio del ristorante in funzione dalla distanza dalla stazione centrale

Seguendo il percorso proposto dal sistema l'utente ha potuto esplorare i dati in maniera efficiente ed automatica. In questo modo egli ha avuto modo di analizzare tutti gli attributi interessanti ai fini dell'organizzazione della sua gita.

3.2.3 Test esterno

Questo test ha lo scopo di valutare, coinvolgendo persone estranee alle strategie utilizzate negli algoritmi, due aspetti relativi al nostro lavoro: la scelta dei parametri dell'analisi statica per una data applicazione per quanto riguarda la selezione della migliore visualizzazione, e i pesi utilizzati per la generazione di un percorso di visualizzazioni.
Per condurre questo test abbiamo creato un questionario on-line in cui abbiamo descritto i due casi d'uso e abbiamo posto alcune domande agli utenti per verificare i risultati che i nostri algoritmi producono. Abbiamo poi deciso anche di chiedere agli utenti se preferiscono avere delle visualizzazioni chiare e semplici o più informative anche se più complesse. Quest'ultima parte non ha l'obiettivo di validare i risultati che producono i nostri algoritmi ma di indirizzare possibili sviluppi futuri verso la generazione di visualizzazioni più semplici o più complesse.
Nel questionario abbiamo descritto due casi d'uso del nostro prototipo rispettivamente per i due scenari presentati nel paragrafo 3.2.1.

Per costruire il questionario abbiamo usato un tool on-line [2] che permette, utilizzando anche un account gratuito, di creare sondaggi e renderli facilmente accessibili ad altre persone. Questo strumento assiste l'utente nella fase di creazione del questionario fornendo una semplice ed intuitiva interfaccia attraverso la quale è possibile scrivere le domande. Dopo aver creato il sondaggio l'utente amministratore del questionario può accedere al back-end per visualizzare le risposte che vengono date dalle persone. Tale back-end permette di visualizzare non solo le varie risposte che sono state date alle domande ma anche varie statistiche più approfondite.

Test scelta della migliore visualizzazione Per ognuno dei due scenari abbiamo inizialmente presentato all'utente una serie visualizzazioni e gli abbiamo chiesto di selezionare quella che a suo parere fosse la migliore.

[2] www.qualtrics.com

Per ognuno dei due casi d'uso, all'utente è stato chiesto di scegliere la visualizzazione che preferiva tra un insieme di rappresentazioni proposte:

- Per il primo scenario sono stati mostrati, una mappa che mostra gli appartamenti, una mappa che mostra le università, una tabella che riassume alcune informazione delle università e un grafico a dispersione che mostra la percentuale di studenti immatricolati in funzione del rapporto medio studenti/facoltà.

- Per il secondo scenario sono stati mostrati, una mappa dei ristoranti, un grafico a dispersione che mostra la distanza per raggiungere il museo dalla stazione centrale in funzione del tempo per raggiungere il museo dalla stazione centrale, un grafico a dispersione che mostra il prezzo medio del ristorante in funzione del tempo per raggiungerlo dalla stazione centrale, una mappa degli hotel e una mappa dei musei.

Test generazione dei percorsi In questa sezione del questionario abbiamo mostrato all'utente le visualizzazioni che verrebbero realmente mostrate in un percorso e gli abbiamo chiesto di ordinarle in base a ciò che sia aspetterebbe in un analogo caso reale.
L'utente doveva ordinare le seguenti visualizzazioni: una mappa dei musei, un diagramma cartesiano che mostrava il prezzo del biglietto del museo in funzione della distanza dalla stazione e una tabella che mostrava varie informazioni sui musei.

Test complessità delle visualizzazioni In questa sezione del questionario abbiamo mostrato all'utente due tipi di visualizzazioni, una semplice, che mostra attributi di una sola entità dello scenario e una più complessa che mostra attributi di più entità coinvolte nello scenario. Gli abbiamo chiesto di selezionare quale fosse il tipo di visualizzazione che preferiva.
Per ognuno dei due scenari è stato mostrato un insieme di visualizzazioni composto da mappe semplici che mostrano le istanze di una sola entità e una

mappa che mostra insieme le istanze di tutte le entità.

In generale per non influenzare la risposta dell'utente abbiamo mischiato l'ordine delle visualizzazioni.
Alla fine del questionario abbiamo posto agli utenti alcune domande per capire il livello di familiarità con aspetti connessi all'uso di un motore di ricerca. In particolare abbiamo posto le seguenti domande:

- sesso
- fascia di età
- livello di conoscenze informatiche (alto, medio, basso)
- familiarità con l'uso dei motori di ricerca (alto, medio, basso)
- familiarità con gli acquisti on-line (alto, medio, basso)
- e-mail

Risultati del test

In questa sezione vengono commentati i risultati che sono stati ottenuti sottoponendo il questionario agli utenti.

Caratterizzazione degli utenti Gli utenti che hanno risposto al questionario sono stati 31 e hanno le seguenti caratteristiche:

- **Sesso** il 76% è di sesso maschile mentre il 24% femminile;
- **Fasce di età** il 34% ha un'età compresa tra 21 e 25 anni, il 41% tra 26 e 30, il 3% tra 31 e 35, mentre il 21% ha un'età superiore a 40 anni;
- **Livello di conoscenze informatiche** il 10% degli utenti ha dichiarato di avere un basso livello di conoscenze in questo campo, il 21% ha dichiarato un livello medio, mentre il 69% ha un livello alto;
- **Familiarità con l'uso dei motori di ricerca** il 14% ha dichiarato un livello basso, il 10% medio, mentre il restante 76% alto;

- **Familiarità con gli acquisti on-line** il 17% ha un livello basso, il 45% medio, mentre il 38% alto.

Test scelta della migliore visualizzazione Per quanto riguarda le due domande del test che mirano a validare l'algoritmo di generazione della migliore visualizzazione, per entrambi gli scenari la maggior parte (per il primo scenario il 71%, mentre per il secondo 74% (figure 3.9-3.10)) degli utenti ha scelto come visualizzaziome migliore quella generata per prima dall'algoritmo (per il primo e il secondo scenario rispettivamente la mappa delle università e la mappa dei musei).
Questo significa che l'algoritmo, analizzando il result-set e i parametri dell'analisi statica, riesce a generare una visualizzazione che, in buona parte dei casi, risulta essere compatibile con le attese dell'utente finale.

Answer		%
Una mappa che mostra gli appartamenti. Il colore dei simboli utilizzati indica il numero di camere da letto di cui ogni appartamento dispone mentre la grandezza è proporzionale al canone d'affitto mensile		10%
Una mappa che mostra le università. Il colore dei simboli utilizzati indica se l'istituto è privato o pubblico mentre la grandezza è proporzionale all'importo delle tasse universitarie		71%
La seguente tabella che mostra i seguenti dettagli delle università		19%
Un grafico cartesiano che mostra le università, in particolare sugli assi troviamo la percentuale di studenti immatricolati e il rapporti medio di studenti per facoltà, mentre il colore indica se l'università è pubblica o privata		0%

Figura 3.9: Scelta visualizzazione migliore scenario università

Answer		%
Una mappa dei musei di Roma, nella quale il colore dei simboli utilizzati indica se il museo considerato è attrezzato o meno per accogliere persone disabili, mentre la loro dimensione è proporzionale alla distanza dalla stazione Centrale		74%
Un grafico cartesiano che mostra i musei, in particolare visualizza il tempo necessario per raggiungerli dalla stazione Centrale in funzione della loro distanza. Inoltre il colore del simbolo di ogni museo indica se esso è attrezzato o meno per accogliere persone disabili		0%
Un grafico cartesiano che mostra i ristoranti, in particolare visualizza il prezzo medio di un pasto in funzione del tempo necessario per raggiungerli dalla stazione Centrale. Inoltre il colore dei simboli utilizzati indica se il ristorante accetta o meno carte di credito		3%
Una mappa degli Hotel della città di Roma, in particolare il colore dei simboli utilizzati indica se l'albergo considerato dispone di parcheggio privato, mentre la loro grandezza è proporzionale alla distanza dalla stazione Centrale		23%
Una mappa dei ristoranti di Roma, in particolare il colore dei simboli utilizzati indica se il ristorante considerato accetta o meno carte di credito, mentre la loro grandezza è proporzionale al prezzo medio di un pasto		0%

Figura 3.10: Scelta visualizzazione migliore scenario Roma

Test generazione dei percorsi Per quanto riguarda la scelta della prima visualizzazione, la maggior parte degli utenti (80%) che hanno partecipato al test, hanno scelto la visualizzazione che l'algoritmo selezionava effettivamente come prima (la mappa che mostra i musei).
Mentre, per quanto concerne la scelta delle due visualizzazioni successive del percorso, si è verificata una situazione di parità. Entrambe le visualizzazioni proposte (il grafico a dispersione e la tabella riassuntiva che rispettivamente l'algoritmo propone al secondo e terzo posto) sono state poste in seconda posizione sostanzialmente dallo stesso numero di utenti (42% e 45%).
Per commentare questo fenomeno dobbiamo considerare il fatto che entrambe le visualizzazioni hanno un alto potere informativo, mostrano delle caratteristiche molto specifiche e richiedono da parte dell'utente un diverso sforzo cognitivo per la loro interpretazione. Di conseguenza la preferenza dell'una piuttosto che dell'altra dipende da molti fattori tra i quali, la familiarità con interfacce complesse e le preferenze personali dell'utente finale.
Per quanto riguarda invece la scelta della visualizzazione da porre come terza, il grafico è stato scelto dal 52% degli utenti, mentre la tabella dal 38%.

Answer	1	2	3
Un grafico cartesiano che mostra i musei, in particolare visualizza il prezzo del biglietto d'ingresso in funzione della distanza dalla stazione Centrale. Inoltre il colore indica se il museo è attrezzato per accogliere le persone disabili	1	13	17
Una mappa dei musei, in particolare il colore dei simboli utilizzati indica se il museo è attrezzato per accogliere persone disabili, mentre la dimensione è proporzionale alla distanza dalla stazione Centrale	25	4	2
La seguente tabella che mostra le seguente informazioni riguardo i musei	5	14	12

Figura 3.11: Percorsi di visualizzazioni

Per effettuare un'analisi più quantitava dei risultati abbiamo scelto di utilizzare il coefficiente di correlazione per ranghi di Spearman [13]. Questo coefficiente indica quanto due sequenze sono correlate tra di loro, e assume valori compresi tra -1 e +1.
In particolare il segno di questo valore indica la direzione della correlazione, mentre la il valore assoluto è propozionale "all'intensità" della correlazione.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{3.1}$$

Per applicare questa formula abbiamo considerato tutte le possibili combinazioni delle tre visualizzazioni proposte per il percorso. Abbiamo quindi contato, per ogni combinazione, il numero di utenti che le hanno selezionate. Per ogni combinazione abbiamo poi calcolato il coefficiente di Spearman rispetto alla sequenza generata dal nostro algoritmo.
A questo punto è stata calcolata la media (μ) pesata rispetto alla percentuale di preferenze e la varianza (σ^2) dei coefficienti ottenuti. Abbiamo poi identificato gli outlier selezionando tutti i coefficienti che non erano compresi nell'intervallo $\mu \pm 2\sigma^2$. Media e varianza sono state poi ricalcolate senza prendere in considerazione gli outlier (che erano 4 su 31 campioni).

Eseguendo i calcoli appena descritti utilizzando un foglio di calcolo, abbiamo ottenuto un valore di ρ pari a 0.759 con una varianza di 0.062.
I valori ottenuti indicano principamente due cose:

- la presenza di una correlazione positiva tra la sequenze scelte e quella generata dall'algoritmo;

- la presenza, in media, di una forte correlazione tra le sequenze indicate dall'utente e quella generata dall'algoritmo in quanto il valore assoluto è maggiore di 0,5.

Test complessità delle visualizzazioni Dai dati raccolti è risultato che generalmente gli utenti preferiscono avere delle visualizzazioni più ricche (figure 3.12-3.13), che mostrino le caratteristiche principali di più di un'entità per volta nonostante in questo modo vengano generate delle visualizzazioni più complesse.
Questo fatto molto probabilmente accade in quanto, essendo i risultati ottenuti da una ricerca multi-dominio, l'utente vuole avere una visione più ampia sui dati visualizzando gli attributi di più oggetti coinvolti dalla ricerca.

Answer		%
Una mappa che mostra gli appartamenti. Il colore dei simboli utilizzati indica il numero di camere da letto di cui ogni appartamento dispone mentre la grandezza è proporzionale al canone d'affitto mensile		6%
Una mappa che riassume le varie informazioni riguardanti sia le università che gli appartamenti		65%
Una mappa che mostra le università. Il colore dei simboli utilizzati indica se l'istituto è privato o pubblico mentre la grandezza è proporzionale all'importo delle tasse universitarie		29%

Figura 3.12: Preferenze complessità visualizzazioni scenario Università

Answer		%
Una mappa dei musei di Roma, nella quale il colore dei simboli utilizzati indica se il museo considerato è attrezzato o meno per accogliere persone disabili, mentre la loro dimensione è proporzionale alla distanza dalla stazione Centrale		32%
La seguente mappa che aggrega le varie informazioni di musei, ristoranti e hotel		55%
Una mappa dei ristoranti di Roma, in particolare il colore dei simboli utilizzati indica se il ristorante considerato accetta o meno carte di credito, mentre la loro grandezza è proporzionale al prezzo medio di un pasto		3%
Una mappa degli Hotel della città di Roma, in particolare il colore dei simboli utilizzati indica se l'albergo considerato dispone di parcheggio privato, mentre la loro grandezza è proporzionale alla distanza dalla stazione Centrale		10%

Figura 3.13: Preferenze complessità visualizzazioni scenario Roma

Capitolo 4

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stata proposta una soluzione che permette di assistere l'utente durante la visualizzazione dei dati che vengono restituiti da un sistema di ricerca Web multi-dominio.
Viene inoltre proposto un metodo per suggerire un percorso che guidi l'utente nell'esplorazione dei risultati.
Per raggiungere questi scopi sono stati sviluppati due algoritmi basati su euristiche che analizzano sia la struttura e il tipo dei dati risultati, sia le caratteristiche statistiche della distribuzione dei loro valori.
Infine gli algoritmi sono stati validati, sia con un test interno che è stato effettuato durante e dopo la fase di implementazione, sia tramite un test esterno che ha coinvolto persone estranee alle strategie utilizzate per l'implementazione degli algoritmi utilizzati.

4.1 Esperienza e discussione

In questo paragrafo illustriamo alcune considerazioni emerse durante il lavoro che abbiamo svolto.

La prima considerazione è nata durante lo sviluppo dell'algoritmo che genera i percorsi di visualizzazioni da proporre agli utenti. Ci siamo accorti che potrebbe essere abbastanza tedioso per l'utente navigare un percorso per più di tre o quattro passi consecutivi. E' invece molto probabile che già du-

rante i primi passi l'utente selezioni uno o più particolari istanze degli oggetti mostrati per generare una visualizzazione innestata.
Abbiamo quindi deciso di fermare l'algoritmo nel momento in cui, dopo aver proposto le prime viste sui dati del result-set, si presentino delle visualizzazioni con un valore del parametro rank assoluto troppo basso.

La questione più importante che abbiamo osservato è che a volte gli algoritmi sviluppati generano delle visualizzazioni nelle quali vengono mostrati attributi che non sempre sono correlati tra loro. Questo fenomeno potrebbe verificarsi soprattutto nelle visualizzazioni con rank basso in quanto nelle prime che vengono proposte sono mostrati attributi molto significativi e quindi è quasi sempre utile mostrarli nella stessa rappresentazione. Invece, nelle visualizzazioni successive alle prime abbiamo attributi meno importanti ma molto specifici e quindi probabilmente scorrelati tra loro.
Per esempio è capitato, all'interno di un percorso nell'ambito dello scenario delle università, che venisse proposta al nono passo dell'algoritmo una visualizzazione mostrante un grafico a dispersione con i seguenti attributi associati ai primi due assi: un indice che quantifica quanto in un campus universitario è possibile muoversi a piedi senza utilizzare mezzi di trasporto sulle ascisse e la percentuale di studenti dell'università considerata che riceve aiuti economici per pagare gli studi sulle ordinate. Questi attributi potrebbero essere interessanti se presi singolarmente, ma potrebbe non avere molto senso mostrarli l'uno in funzione dell'altro.
Questo problema è causato dal fatto che tuttora non vi è alcun modo di inferire, nè dai dati dell'analisi statica nè tantomeno dai parametri calcolati nell'analisi dinamica, un grado di correlazione tra due attributi.

Un'altra questione emersa è relativa a quegli attributi che indicano caratteristiche molto specifiche di una certa entità. L'importanza di tali attributi è molto soggettiva. Per questo motivo, per una certa applicazione potrebbe essere molto importante mostrare che un hotel disponga del minibar in ogni camera, mentre per un'altra applicazione questo potrebbe essere totalmente irrilevante, rispetto per esempio al fatto di visualizzare la presenza di rete

wireless. Per fare un esempio più concreto potremmo dire che per un'applicazione che permette di organizzare vacanze è molto importante indicare la presenza del minibar, mentre per un'applicazione che permette agli utenti di organizzare un soggiorno per partecipare ad una conferenza sarebbe molto più importante indicare la disponibilità di una rete wireless.
Quindi, per quanto riguarda questo tipo di attributi, le loro probabilità di essere mostrati nelle visualizzazioni devono essere impostate calibrando i parametri dell'analisi statica. Questo può essere fatto sfruttando la possibilità di creare un'apposita configurazione per un certo attributo come illustrato nel paragrafo 2.1.1.

4.2 Sviluppi futuri

Lavori futuri si potrebbero concentrare su di un ulteriore raffinamento degli algoritmi. In particolare un possibile miglioramento potrebbe essere volto a cercare di risolvere il problema della correlazione degli attributi esposto nelle considerazioni.
Una possibile soluzione potrebbe consistere nell'introduzione di una matrice di correlazione degli attributi. Quest'ultima sarà configurabile da parte "dell'Expert User", il quale potrà inserire dei valori che quantifichino, per ogni attributo, quanto esso sia correlato con gli altri.
Un'ipotetica matrice del genere sarebbe molto utile nell'algoritmo di generazione della migliore visualizzazione nella fase in cui vengono scelti i singoli attributi da mostrare. Una volta scelto il primo attributo da assegnare a una certa dimensione di una visualizzazione, la classifica dei rank dei possibili attributi associabili alle altre dimensioni dovrà essere fortemente influenzato dal relativo valore di correlazione.
D'altra parte però bisogna tenere conto del fatto che una matrice di correlazione come quella descritta dovrebbe contenere molti valori. In particolare, in un generico scenario, essa conterrà $\frac{n^2}{2}$ valori dove n è il numero degli attributi totali di tutti gli oggetti coinvolti dall'applicazione considerata. Il numero di valori corrisponde al quadrato di n diviso per due in quanto la

matrice di correlazione è simmetrica.
Ogni volta in cui un “Expert User” volesse sviluppare una nuova applicazione dovrebbe quindi riempire una matrice molto grande, questo andrebbe in contrasto con l'obiettivo che ci eravamo posti quando abbiamo introdotto il concetto di tipi semantici.
Non è possibile, però, creare questa matrice utilizzando i tipi semantici in quanto il concetto di correlazione è strettamente connesso all'applicazione e al singolo attributo.

Un altro possibile sviluppo sarebbe quello di introdurre l'analisi delle statistiche di utilizzo degli utenti.
Si potrebbe tener traccia delle interazioni dell'utente con l'interfaccia volte a cambiare gli attributi che vengono scelti dagli algoritmi per essere mostrati sulle varie dimensioni di una certa visualizzazione.
Così facendo, si potrebbero modificare gli algoritmi in modo tale che possano, durante la generazione delle viste, tener conto delle statistiche registrate.
Questo cambiamento permetterà di introdurre negli algoritmi il concetto di apprendimento e di alleggerire il lavoro di raffinamento dei parametri dell'analisi statica da parte ”dell'Expert User”.

Bibliografia

[1] Stefano Ceri, Search Computing Infrastructure, Deliverable R2 of the Search Computing (SeCo) Project

[2] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Piero Fraternali, Liquid Query: Multi-Domain Exploratory Search on the Web

[3] Marco Brambilla, Alessandro Campi, Stefano Ceri, and Silvia Quarteroni, Semantic Resource Framework

[4] Jose F. Rodrigues Jr., Agma J. M. Traina, Maria Cristina F. de Oliveira, Caetano Traina Jr., Reviewing Data Visualization: an Analytical Taxonomical Study

[5] Ed H. Chi, Xerox Palo Alto Research Center, A Taxonomy of Visualization Techniques using the Data State Reference Model

[6] Tiziana Catarci, Giuseppe Santucci, Maria Costabile, Foundations of the DARE system for Drawing Adequate REpresentation

[7] Chris Stolte, Diane Tang, and Pat Hanrahan, Polaris: A System for Query, Analysis, and Visualization of Multidimensional Databases

[8] Marco Brambilla, Alessandro Campi, Stefano Ceri, and Silvia Quarteroni, Semantic Resource Framework

[9] Anand Rajaraman, Kosmix: High-Performance Topic Exploration using the Deep Web

[10] Gary Marchionini, Exploratory search: from finding to understanding

[11] Michelle Q Wang Baldonado and Terry Winograd, SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests

[12] Ghigini Federico, Luzzara Nicola, Minelli Marco Giuseppe, Vit Tiziano, Utilizzo di algoritmi per l'esplorazione di spazi da parte di robot in applicazioni information-retrieval, Politecnico di Milano

[13] Spearman's rank correlation coefficient, http://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient